Customer Notification

V850™Series

CPDW9X/NT-CDR-V85X, Y-GHS-MULTI-V800

Operating Precautions

GHS MULTI 2000 Integrated Development Environment Version 3.X, 4.X,5.X,6.X

Renesas Electronics
www.renesas.com

Document No. r20tu0003ed1812
Date Published: 14/09/16

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.

2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.

4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.

5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.

6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.

7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti- crime systems; safety equipment; and medical equipment not specifically designed for life support.
   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.

8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.

9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.

10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.

12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority- owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

Table of Contents

(A) Table of Operating Precautions 5

(B) Description of Operating Precautions 13

(C) Valid Specification 115

(D) Revision History 117

(A) Table of Operating Precautions

No.	Outline / Version	CPDW9X/NT-CDR-V85x Y-GHS-MULTI-V800								
	Version	4.0.7a	4.2.3	4.2.4	5.1.6C	5.1.7D	5.3.0	6.1.4/ 2012.5.1	6.1.4/ 2013.1.5	6.1.4/ 2013.5.5
a1	Multi Debugger displays incorrect values for PSW flags	✓	✓	✓	✓	✓	✓	✓	✓	✓
a2	Compiler generates incorrect volatile access	✓	✓	✓	✓	✓	✓	✓	✓	✓
a3	Compiler generates incorrect branches inside an interrupt services routine	✓	✓	✓	✓	✓	✓	✓	✓	✓
a4	Using PIC, the compiler generates incorrect addresses for function in different sections	✓	✓	✓	✓	✓	✓	✓	✓	✓
a5	Compiler stops with an internal error by using section renaming in combination with “.bss” and “.sbss” sections	✓	✓	✓	✓	✓	✓	✓	✓	✓
a6	Compiler uses library function instead of “mulh” instruction	✓	✓	✓	✓	✓	✓	✓	✓	✓
a7	Two-pass inlining fails with new generation compiler	✓	✓	✓	✓	✓	✓	✓	✓	✓
a8	Compiler generates incorrect code for exclusive or operation.	✓	✓	✓	✓	✓	✓	✓	✓	✓
a9	Two-pass inlining and section renaming causes incorrect linker behavior	✓	✓	✓	✓	✓	✓	✓	✓	✓
a10	Wrong debug information are generated involving the peephole optimizer	✓	✓	✓	✓	✓	✓	✓	✓	✓
a11	Compiler ignores section renaming in combination with the inline assembly option	✓	✓	✓	✓	✓	✓	✓	✓	✓
a12	Multi Debugger displays incorrect values for the CTPSW and CTPC registers	✓	✓	✓	✓	✓	✓	✓	✓	✓
a13	The “#pragma intvect” directive fails with new generation compiler	✘	✘	✘	✓	✓	✓	✓	✓	✓

No.	Outline	CPDW9X/NT-CDR-V85x Y-GHS-MULTI-V800								
	Version	4.0.7a	4.2.3	4.2.4	5.1.6C	5.1.7D	5.3.0	6.1.4/ 2012.5.1	6.1.4/ 2013.1.5	6.1.4/ 2013.5.5
a14	Compiler generates incorrect code for infinite loop	✓	✓	✓	✓	✓	✓	✓	✓	✓
a15	Compiler generates incorrect code for a loop followed by a “#pragma asm” construct	✓	✓	✓	✓	✓	✓	✓	✓	✓
a16	Compiler stops with internal error	✓	✓	✓	✓	✓	✓	✓	✓	✓
a17	The element pointer will be destroyed within interrupt service routine	✓	✓	✓	✓	✓	✓	✓	✓	✓
a18	SFR target command blocks the target window	✓	✓	✓	✓	✓	✓	✓	✓	✓
a19	Incorrect macro expansion on ld.bu instruction	✓	✓	✓	✓	✓	✓	✓	✓	✓
a20	Incorrect behavior on delete function optimization option "-OD"	✓	✓	✓	✓	✓	✓	✓	✓	✓
a21	Loop optimization on volatile access generates endless loop	✓	✓	✓	✓	✓	✓	✓	✓	✓
a22	Unresolved static functions after two path inlining	✓	✓	✓	✓	✓	✓	✓	✓	✓
a23	Automatically open editor on errors	✓	✓	✓	✓	✓	✓	✓	✓	✓
a24	GNUGREP generates application error	✓	✓	✓	✓	✓	✓	✓	✓	✓
a25	Multiply defined type identifier on virtual functions	✓	✓	✓	✓	✓	✓	✓	✓	✓
a26	"-no_callt" option caused wrong restore value for register R2 within ISR	✓	✓	✓	✓	✓	✓	✓	✓	✓
a27	Incorrect code generation on bit accesses	✓	✓	✓	✓	✓	✓	✓	✓	✓
a28	Bad epilogue in interrupt with “-prepare_dispose” and “-notda”	✓	✓	✓	✓	✓	✓	✓	✓	✓
a29	Wrong optimization for "Common-subexpr elimination"	✓	✓	✓	✓	✓	✓	✓	✓	✓
a30	Wrong address calculation	✓	✓	✓	✓	✓	✓	✓	✓	✓

No.	Outline	CPDW9X/NT-CDR-V85x Y-GHS-MULTI-V800								
	Version	4.0.7a	4.2.3	4.2.4	5.1.6C	5.1.7D	5.3.0	6.1.4/ 2012.5.1	6.1.4/ 2013.1.5	6.1.4/ 2013.5.5
a31	Wrong memory access optimization on SFR's volatile variables.	✓	✓	✓	✓	✓	✓	✓	✓	✓
a32	Syntax error on Rept inside a macro	✓	✓	✓	✓	✓	✓	✓	✓	✓
a33	Use the bit instruction NOT1 is not systematic in XOR operation	✓	✓	✓	✓	✓	✓	✓	✓	✓
a34	Debug command print fails if a variable as the same name as one of the command option	✓	✓	✓	✓	✓	✓	✓	✓	✓
a35	Compiler option –nomacro is ignored when used with –cpu=?	✓	✓	✓	✓	✓	✓	✓	✓	✓
a36	EP is not restored in ISR that calls func_ptr	✓	✓	✓	✓	✓	✓	✓	✓	✓
a37	The debug command copy has a size limit	✓	✓	✓	✓	✓	✓	✓	✓	✓
a38	tda-offset out of range in sld.bu instruction	✓	✓	✓	✓	✓	✓	✓	✓	✓
a39	In case of redundant load, an unsigned load is considered unnessary in case a signed load to the same location was previously done.	✓	✓	✓	✓	✓	✓	✓	✓	✓
a40	In the expression, int a %=(1+~(unsigned b)), only the % is done.	✓	✓	✓	✓	✓	✓	✓	✓	✓
a41	Assignment to a bit field element gives an Internal Compiler Error (ICE) when compiled with the option –Ospace	✓	✓	✓	✓	✓	✓	✓	✓	✓
a42	When compiled with –Ospace, an assignment to a structure's element will also set to 0 all other elements	✓	✓	✓	✓	✓	✓	✓	✓	✓
a43	In the expression, n=s.m++ & s.m++, the incrementation is done only once.	✓	✓	✓	✓	✓	✓	✓	✓	✓

No.	Outline	CPDW9X/NT-CDR-V85x Y-GHS-MULTI-V800								
	Version	4.0.7a	4.2.3	4.2.4	5.1.6C	5.1.7D	5.3.0	6.1.4/ 2012.5.1	6.1.4/ 2013.1.5	6.1.4/ 2013.5.5
a44	When compiling with –Ospace, the operator | is replaced with ||. This implies that the second expression is skipped if the first one is true.	✓	✓	✓	✓	✓	✓	✓	✓	✓
a45	The environment affects the code generated.	✓	✓	✓	✓	✓	✓	✓	✓	✓
a46	The response time of the debugger is very slow when the I/O window is waiting for an input.	✓	✓	✓	✓	✓	✓	✓	✓	✓
a47	The linker option –codefactor breaks the code in interrupt functions when code is optimized for speed	✓	✓	✓	✓	✓	✓	✓	✓	✓
a48	Any macro containing a sequence with a '.' followed by other characters is not expanded correctly.	✓	✓	✓	✓	✓	✓	✓	✓	✓
a49	Short Load instruction patch failure	✓	✓	✓	✓	✓	✓	✓	✓	✓
a50	r20/r21 used by -codefactor incorrectly	✓	✓	✓	✓	✓	✓	✓	✓	✓
a51	The linker gives an application error when the option -gnu_linkmap is passed to the compiler driver.	✗	✗	✗	✓	✓	✓	✓	✓	✓
a52	V850e1f (Floating Point Unit): Wrong code generation for floating point comparisons	✓	✓	✓	✓	✓	✓	✓	✓	✓
a53	Single Step Limitation using OCD N-Wire Debugger	✗	✗	✗	✗	✗	✗	✗	✗	✗
a54	Peephole Optimizer does not generate proper code	✓	✓	✓	✓	✓	✓	✓	✓	✓
a55	Option '-passsource' may affect code generation	✓	✓	✓	✓	✓	✓	✓	✓	✓
a56	The linker creates only one far_call patch function per label	✓	✓	✓	✓	✓	✓	✓	✓	✓

No.	Outline	CPDW9X/NT-CDR-V85x Y-GHS-MULTI-V800								
	Version	4.0.7a	4.2.3	4.2.4	5.1.6C	5.1.7D	5.3.0	6.1.4/ 2012.5.1	6.1.4/ 2013.1.5	6.1.4/ 2013.5.5
a57	The Multi Debugger displays large volatile structures very slowly	×	×	×	✓	✓	✓	✓	✓	✓
a58	Compiler seg-fault with –farcalls –pic –mtda	✓	✓	✓	✓	✓	✓	✓	✓	✓
a59	Incorrect conditional branch after tst1	✓	✓	✓	✓	✓	✓	✓	✓	✓
a60	A double constant might be converted to long long.	✓	✓	✓	✓	✓	✓	✓	✓	✓
a61	Far call patch of interrupt prologue function overwrites R1	✓	✓	✓	✓	✓	✓	✓	✓	✓
a62	16-bit multiply and divide are used instead of 32-bit operations	✓	✓	✓	✓	✓	✓	✓	✓	✓
a63	Linker does not delete unused functions	✓	✓	✓	✓	✓	✓	✓	✓	✓
a64	Utility gsrec, ghexfile fails padding, if ROM access is split to even/odd.	✓	✓	✓	✓	✓	✓	✓	✓	✓
a65	Make Dependency Generator creates too much dependencies	✓	✓	✓	✓	✓	✓	✓	✓	✓
a66	The compiler dereferenced the wrong register in certain places due to an incorrect optimization.	✓	✓	✓	✓	✓	✓	✓	✓	✓
a67	The compiler generates different object code for identical source modules.	×	✓	✓	✓	✓	✓	✓	✓	✓
a68	The FPU library for V850E1F contains “callt” even if omitted by the compiler	✓	✓	✓	✓	✓	✓	✓	✓	✓
a69	Out-Of-Order Code Generation	×	✓	✓	✓	✓	✓	✓	✓	✓
a70	Incorrect Warning with ‘#pragma ghs io’	-	×	×	✓	✓	✓	✓	✓	✓
a71	Spaces not recognized by elxr in @files	-	×	×	✓	✓	✓	✓	✓	✓
a72	Comments not recognized by elxr in @files	-	×	✓	✓	✓	✓	✓	✓	✓
a73	Wrong Default Libray Selection	✓	×	✓	✓	✓	✓	✓	✓	✓

No.	Outline	CPDW9X/NT-CDR-V85x Y-GHS-MULTI-V800								
	Version	4.0.7a	4.2.3	4.2.4	5.1.6C	5.1.7D	5.3.0	6.1.4/ 2012.5.1	6.1.4/ 2013.1.5	6.1.4/ 2013.5.5
a74	Wrong interpretation of option '-T'	×	×	✓	✓	✓	✓	✓	✓	✓
a75	Internal Error with '-OI'	×	×	✓	✓	✓	✓	✓	✓	✓
a76	Prevent use of set1 on volatile accesses other than char	×	×	✓	✓	✓	✓	✓	✓	✓
a77	Volatile Initializer forgotten	×	×	✓	✓	✓	✓	✓	✓	✓
a78	The compiler does not accept ‚float' – parameters for assembler macros	-	×	×	×	×	×	×	×	×
a79	The compiler emitts the instruction "signed load", instead of "unsigned load"	×	×	✓	✓	✓	✓	✓	✓	✓
a80	Wrong access order to 'volatile' arrays	×	×	✓	✓	✓	✓	✓	✓	✓
a81	Missing Compiled Line	-	×	×	✓	✓	✓	✓	✓	✓
a82	New compiler option	✓	✓	✓	×	×	✓	✓	✓	✓
a83	Support Of Bigger Trace using IECUBE2	✓	✓	✓	×	×	✓	✓	✓	✓
a84	gsrec is missing last segaddr record	✓	✓	✓	×	✓	✓	✓	✓	✓
a85	Internal error of linker elxr	✓	✓	✓	×	✓	✓	✓	✓	✓
a86	Gasmlist fails to display prolog and epilog	✓	✓	✓	×	✓	✓	✓	✓	✓
a87	Compiler does not generate ".rozdata" section	✓	✓	✓	×	✓	✓	✓	✓	✓
a88	Assembler cannot jump to absolute address	✓	✓	✓	×	✓	✓	✓	✓	✓
a89	GHS target agent for external flash loading and memory test fails	✓	✓	✓	×	×	✓	✓	✓	✓
a90	Internal CodeFactor Error	✓	✓	✓	×	✓	✓	✓	✓	✓
a91	Delayed/broken start of MULTI IDE	✓	✓	✓	×	×	✓	✓	✓	✓
a92	Option '-list' influenced code optimizer	×	×	×	×	×	✓	✓	✓	✓

No.	Outline	CPDW9X/NT-CDR-V85x Y-GHS-MULTI-V800								
	Version	4.0.7a	4.2.3	4.2.4	5.1.6C	5.1.7D	5.3.0	6.1.4/ 2012.5.1	6.1.4/ 2013.1.5	6.1.4/ 2013.5.5
a93	Option '-misalign_pack' generates optimized target access	✓	✓	✓	×	×	✓	✓	✓	✓
a94	ELXR internal error	✓	✓	✓	×	×	✓	✓	✓	✓
a95	ELXR internal error with option '-v'	✓	✓	✓	×	×	✓	✓	✓	✓
a96	Simulator is changed for SCH1L instruction simulation	✓	✓	✓	×	×	✓	✓	✓	✓
a97	Missing Compiled Line	✓	✓	×	✓	✓	✓	✓	✓	✓
a98	ELXR generates different output with option '-Olink'	✓	✓	✓	×	×	×	×	×	×
a99	Address Assignment set to 0 with pragma ghs io	✓	✓	✓	✓	×	✓	✓	✓	✓
a100	V61.4 release with incomplete E1 debugger support	✓	✓	✓	✓	✓	✓	×	×	×
a101	Internal Compiler Error "delptr"	✓	✓	✓	✓	✓	✓	×	×	×
a102	Bad short load/store optimization	✓	✓	✓	✓	✓	✓	×	×	×
a103	Size optimization of bitfields	✓	✓	✓	✓	✓	✓	×	×	×
a104	Compiler may hang	✓	✓	✓	✓	✓	✓	×	×	×
a105	Error in gsrec	✓	✓	✓	×	×	✓	✓	✓	✓
a106	Linker does not delete unused functions	✓	✓	✓	✓	✓	✓	×	×	×
a107	Problem with hex constants in ease850	✓	✓	✓	✓	✓	✓	×	×	×
a108	Compiler interrupt routines do not save FPSR and FPEPC	✓	✓	✓	✓	✓	✓	×	×	×
a109	Linker invokes unknown section SHN_ABS	✓	✓	✓	✓	✓	✓	×	×	×
a110	Multi IDE crash in New Project Wizzard with MultiCoreArchive projects	-	-	-	-	-	-	-	-	×

No.	Outline	CPDW9X/NT-CDR-V85x Y-GHS-MULTI-V800								
	Version	4.0.7a	4.2.3	4.2.4	5.1.6C	5.1.7D	5.3.0	6.1.4/ 2012.5.1	6.1.4/ 2013.1.5	6.1.4/ 2013.5.5
a111	Invalid bitfield handling in optimized If/else clause	-	-	-	-	✘	✘	✘	✘	✓
a112	P26: TOOLS-20948 write to global variable was wrongly moved before __DI	-	-	-	-	-	-	✘	✘	✘
a113	P23: BTO-2134 fix error in code motion optimization	-	-	-	-	-	-	✘	✘	✘
a114	P26: BTO-2144 update global loop variables in nested loops	-	-	-	-	-	-	✘	✘	✘
a115	P25: BTO-2136/ .ghsexports with library; BTO-2139 exportall should not imply extractall	-	-	-	-	✘	✘	✘	✘	✓
a116	P22: BTO-2095 ease850 now recognizes bt and bf	-	-	-	-	✘	✘	✘	✘	✓
a117	P21: BTO-2089 accurate simulation of ldsr FPST and ldsr FPSFG	-	-	-	-	✘	✘	✘	✘	✓

✓ Not applicable
✘ Applicable
- Not checked

Note: The rank is indicated by the letter appearing at the 5th position from the left in the lot number, marked on each product.

(B) Description of Operating Precautions

No. a1	**Multi Debugger displays incorrect values for PSW flags**
	Version Information *V3.3* *Details* The Multi debugger displays incorrect values for PSW flags. *Workaround* Use the value of whole PSW register instead of single flags.

No. a2	**Compiler generates incorrect volatile access**
	Version Information *V3.3* *Details* A volatile half word memory access, such as: `* (( volatile unsigned short *) 0xffffff010) |=0x80;` is done by using set1 instruction instead of using ld.h / st.h instruction. *Workaround* Use option –Z982 to disable the bit commands. A new compiler, which fixes this problem, can be downloaded from the NEC WEB page(s) at http://www.renesas.eu/updates?id=443

No. a3	**Compiler generates incorrect branches inside an interrupt services routine**

Version Information

V3.3

Details

In object mode the compiler generates incorrect branches inside an interrupt service routine.

Example:

```
extern void foo(void);
extern int test(void);

void isr (void)
{
  #pragma ghs interrupt

  if (test()==1)
   {
    foo();
   }
}
```

This will generate an endless loop in case result of “test()” is not 1.

Workaround

Use option –noobj to use assembly mode for compilation.

No. a4	**Using PIC, the compiler generates incorrect addresses for functions in different sections**

Version Information

V3.3

Details

In case PIC is used, the compiler generates incorrect addresses for functions in different sections and modules.

Example:

a.c:

```
extern void init();
void foo(void){};
void main(void)
{
   init();
}
```

b.c:

```
#pragma ghs section text=".mytext"
volatile int addr;
extern void foo(void);
void init()
{
   addr = (int)foo;
}
```

This will generate an incorrect result for “addr” in function “init()”.

Workaround

Use option –obj to compile in object mode.

A new compiler, which fixes this problem, can be downloaded from the NEC WEB page(s) at http://www.renesas.eu/updates?id=443

No. a5	**Compiler stops with an internal error by using section renaming in combination with ".bss" and ".sbss" sections**

Version Information

V3.3

Details

The compiler stops with "Internal Compiler Error Code (96387)" in case

```
#pragma ghs section bss=".name"
```

or

```
#pragma ghs section sbss=".name"
```

is used.

Example:

```
#pragma ghs section bss=".myseg"
int a;
```

Workaround

Use option –noobj to use assembly mode for compilation.

No. a6	**Compiler uses library function instead of “mulh” instruction**

Version Information

V3.3

Details

The compiler uses library function instead of “mulh” instruction of the V850 core.
Example:

test.c:

```
extern int a,b,c;
void foo(void)
{
   a = ((short)(b>>16))*(short)c;
   a = (a<<16)+(short)b*(short)c;
}
```

Command Sequence

ccv850 –c2000 –S –Ospeed –zda=all test.c

test.s:

```
_foo:
        add     -4,sp
        st.w    lp,0[sp]        --          .bf
        ld.h    zdaoff(_c)[zero],r7
        ld.w    zdaoff(_b)[zero],r8
        mov     r7,r7
        mov     r8,r6
        sar     16,r6
        jarl    ___MUL_S16_32,lp
        st.w    r10,zdaoff(_a)[zero]
        mov     r10,r9
        shl     16,r9
        mov     r8,r6
        shl     16,r6
        sar     16,r6
        mov     r6,r6
        jarl    ___MUL_S16_32,lp
        add     r9,r10          --          .ef
        st.w    r10,zdaoff(_a)[zero]
        ld.w    0[sp],lp
        add     4,sp
        jmp     [lp]
```

Workaround

Use option –compat to compile with the compatibility mode compiler or use option --option=51.

No. a7	**Two-pass inlining fails with the new generation compiler**

Version Information

V3.3

Details

Two-pass inlining fails with the new generation compiler (option -c2000).

Example:

a.c:

```
extern void foo(int x);
void main(void)
{
   foo(1);
}
```

b.c:

```
int counter;
void foo(int x)
{
   counter += x;
}
```

Command Sequence

ccv850 -OI=foo -S a.c b.c

```
"a.c", line 1: error: extern inline function "foo" was referenced but
       not defined

extern void foo(int x);
            ^
```

Workaround

Use option “–compat” to compile with the compatibility mode compiler.

No. a8	**Compiler generates incorrect code for exclusive-or-operation**

Version Information

V3.3

Details

The Compiler generates incorrect code for “exclusive-or-operation” in combination with space optimization –OS and compiler driver option –reserve_r2. The “xori” instruction is not insert by the compiler, because the peephole optimizer does not alter instructions whose condition codes were needed.

Example:

test.c:

```
#define TRUE 1
#define FALSE 0
extern unsigned int foo(unsigned char a)
{
      return(0xA8FF);
}
extern unsigned char test(void)
{
   if( (foo(0)^0xFFFF) & 0x8000) return TRUE;
   return FALSE;
}
```

Command Sequence

ccv850 –list –c -noobj -OS -reserve_r2 v010927a.c

test.s:

```
_test:
         jarl    ___ghssavelpa,r10
         mov     0,r6   --      .bf
         jarl    _foo,lp
         shr     16,r10
         bnl     .L19
         mov     1,r9
         br      .L17
.L19:
         mov     0,r9
.L17:
         mov     r9,r10 --           .ef
         jarl    ___ghsloadlp,lp
```

Workaround

Use option -Onopeep to disable the peephole optimization.

No. a9	**Two-pass inlining and section renaming causes incorrect linker behavior**

Version Information

V3.3

Details

Two-pass inlining and section renaming causes incorrect linker behavior. By inlining function

"wait" the linker places “foo” into the ".data" section, unnoticed of the pragma directive.

Example:
v011012a.c:

```
#pragma ghs section data=".myseg"
static short foo[10000] ={0};
#pragma ghs section data=default

int a=0;

void wait(void)
{
  int i=0;
  while(i!=100)i=i+1;
}

void main(void)
{
    while(a!=10000)
    {
     foo[a] = a;
     wait();
     a++;
    }
}
```

Command Sequence
ccv850e –noobj -compat -c -list -passsource -g -OI=wait test.c
ccv850e –map test.ld test.o -o test

```
[elxr] (error) section .data (0xffffc000-0xe3f) doesn't fit in memory
block Default (0-0xffffffff)
```

Workaround

Use the “__inline” keyword in C code, instead of inlining option –OI.

Example:

```
__inline void wait(void)
{
  int i=0;
  while(i!=100)i=i+1;
}
```

No. a10	**Wrong debug information are generated involving the peephole optimizer**

Version Information

V3.3, V3.5, V4.0.5, V4.0.5c

Details

Using higher optimization options that are involving the peephole optimizer, i.e. -OM, -OS, -OL or –O, causes wrong debug information.
This is a "**limitation"** of the product. In particular, peephole optimizations which do many things to improve code quality and to eliminate instructions happens after debug information is emitted.

For Multi 2000 v3.5.1, please refer also to the "Green Hills C/C++ User´s Guide" chapter 6, "Problems with Source Level Debuggers".
For Multi 2000 v4.0.5, please refer to the "Building Applications for Embedded V800", chapter18, "Problem with source Level Debuggers".

Example:

The local variable "UC" is not in the register location reported by the debugger. In fact, the local variable does not really exist, but the debugger information indicates that it does.

```
unsigned char temp;
void PA(unsigned char a)
{
    temp=a;
}
unsigned char PB(void)
{
    return (temp+1);
}
unsigned char test (unsigned char UD[4])
{
    unsigned char UC;
    UC = UD[4];
    PA(UC);
    UC=PB();
    return (UC);
}
int main ()
{
    unsigned char B [] = "01234567";
    temp = test (B);
    return (temp);
}
```

Workaround

Use option -Onopeep to disable the peephole optimization.

No. a11	**Compiler ignores section renaming in combination with the inline assembly option**

Version Information

V3.3, V3.5, V4.0.5, V4.0.5c

Details

In case of, section renaming and inline assembly is used at the same time, the section renaming directive “#pragma ghs section …” has no effect.

Example:

The assembler code sequence will be placed into the “.text” section instead of the user defined section “.initvec”.

```
#pragma ghs section text=".initvec"
#pragma asm
reset:
          nop
          nop
          nop
          nop
          nop
          jr _start

#pragma endasm
#pragma ghs section text=default
```

Workaround

Use section renaming within the assembler sequence.

```
#pragma asm
.section ".initvec",.text
reset:
        nop
        nop
        nop
        nop
        nop
        jr _start

#pragma endasm
```

No. a12	**Multi Debugger displays incorrect values for the CTPSW and CTPC register**

Version Information

V3.3

Details

Multi Debugger displays incorrect values for the CTPSW and CTPC register.

Workaround

Update the register definition file “850_creg.rdf” of your current Multi installation, path “C:\Green\defaults\registers”.

A new register definition file, which fixes this problem, can be downloaded from the NEC WEB page(s) at http://www.renesas.eu/updates?id=443

No. a13	**The "#pragma intvect" directive fails with new generation compiler**

Version Information

V3.3, V3.5, V4.0.5, V4.0.5c, V4.0.7(a)

Details

The "#pragma intvect" directive fails by using the new generation compiler and the compiler option "-japanese_automotive_c".

Example:

test.c:

```
#pragma intvect test 0x100

volatile int count;
void main(void)
{
    while(1);
}

#pragma ghs interrupt
void test(void)
{
    count++;
}
```

Command Sequence

ccv850e -c2000 -c -noobj -japanese_automotive_c -zda=all test.c

Workaround

Add definition of interrupt service routine in front of the "#pragma intvect" directive.

```
__interrupt void test (void);                /* Add this line */
#pragma intvect test 0x100

volatile int count;
void main(void)
{
     while(1);
}

#pragma ghs interrupt
void test(void)
{
     count++;
}
```

No. a14	**Compiler generates incorrect code for infinite loop**

Version Information

V3.5

Details

Compiler incorrectly removes calls within an infinite loop by using standard optimization option "–O".

Example:

test.c:

```
unsigned short    test(unsigned short, unsigned char);
unsigned short    a[6], b[6];
unsigned char     c;
void main(void)
{
    c    = 0;
    b[3] = 0x8000;
    b[4] = 0x8000;
    b[5] = 0x8000;
    for(;;)
    {
        a[3] = test(b[3], c);
        a[4] = test(b[4], c);
        a[5] = test(b[5], c);
    }
}
```

Command Sequence

ccv850 –list –c -noobj -O test.c

test.s:

```
  _main:
        add     -4,sp
        st.w    lp,0[sp]
        movhi   hi(_c),zero,r10
        st.b    zero,lo(_c)[r10]
        ori     32768,zero,r2
        movhi   hi(_b),zero,r11
        movea   lo(_b),r11,r11
        st.h    r2,6[r11]
        st.h    r2,8[r11]
        andi    65535,r2,r10
        mov     r11,r2
  .L6:
        st.h    r10,10[r2]
        br      .L6
```

Workaround

A new compiler, which fixes this problem, can be downloaded from the NEC WEB page(s) at http://www.renesas.eu/updates?id=443

No. a15	**Compiler generates incorrect code for a loop followed by a "#pragma asm" construct**

Version Information

V3.3, V3.5

Details

Compiler incorrectly in-lines the assembler sequences into the for-next loop.

Example:

```
void foo(int a);
void main(void)
{
    int i;
    for ( i = 0; i<10 ; i++)
    {
        foo(i);
    }
#pragma asm
        stsr PSW,r2
        ori 0x80,r2,r2
        ldsr r2,PSW
#pragma endasm
}
```

Command Sequence

ccv850e -c –noobj -list test.c

test.s:

```
  _main:
          add     -8,sp
          st.w    lp,4[sp]
          st.w    r29,0[sp]
          mov     0,r29
  .L6:
          mov     r29,r6
          jarl    _foo,lp
          stsr PSW,r2
          ori 0x80,r2,r2
          ldsr r2,PSW
          add     1,r29
          cmp     10,r29
          blt     .L6
          ld.w    4[sp],lp
          ld.w    0[sp],r29
          add     8,sp
          jmp     [lp]
```

Workaround

1) Use the "__asm()" statement for each in-line assembly instruction.
2) Insert an empty statement (semicolon (;)) at the end of the for loop compound statement.

Workaround (continued)

3) Use the definition of an “asm” function for the corresponding assembler sequence:

```
extern void foo(int a);
asm void myasm ()
{
          stsr PSW,r2
          ori 0x80,r2,r2
          ldsr r2,PSW
};
void main(void)
{
     int i;

     for ( i = 0; i<10 ; i++)
     {
          foo(i);
     }
     myasm();
}
```

No. a16	**Compiler stops with internal error**

Version Information

V3.5.1

Details

Compiler stops with internal error. The problem occurs if the compiler can use the "set1" instruction on a TDA variable. The bit-wise OR of more then one bit works correct. In this case "ori" instruction can be used.

Example:

test.c:

```
#pragma ghs starttda
char test;
#pragma ghs endtda

void foo(void)
{
      test |= 0x01;
}
```

Command Sequence

ccv850e -noobj -c -g -O test.c

```
"test.c", line 8: Internal Compiler Error
```

Workaround

A new compiler (PATCH to 3.5.1b4), which fixes this problem, can be downloaded from the NEC WEB page(s) at http://www.renesas.eu/updates?id=443

No. a17	**The element pointer will be destroyed within interrupt service routine**

Version Information

V3.5.1

Details

The element pointer will be destroyed within interrupt service routine. The element pointer is used for register storing within the function prologue. After function prologue "__ghs32rsave00i" was executed the EP will not be recovered. The following access to variables in the tiny data area fails.

Example:

test.c:

```
#pragma ghs starttda
volatile unsigned foo;
#pragma ghs endtda


void main(void)
{
    while(1);
}

#pragma ghs interrupt
void timer(void)
{
       foo++;
}
```

Command Sequence

ccv850 –noobj -c -g -list -passsource -no_inline_prologue v021021a.c

Workaround

A new compiler (PATCH to 3.5.1b4), which fixes this problem, can be downloaded from the NEC WEB page(s) at http://www.renesas.eu/updates?id=443

No. a18	**SFR target command blocks the target window**

Version Information

V3.5.1

Details

The target command "sfr" blocks the 850eserv target window. After typing the "sfr" command to view a specific register and pressing the "enter" key without changing the SFR value the command prompt does not appear any more. Any further command input is blocked.

Workaround

A new compiler (PATCH to 3.5.1b4), which fixes this problem, can be downloaded from the NEC WEB page(s) at http://www.renesas.eu/updates?id=443

No. a19	**Incorrect macro expansion on ld.bu instruction**

Version Information

V3.3, V3.5

Details

A wrong base address (register r1) and displacement is generated on the macro expansion for the ld.bu instruction.

Example:

test.850:

```
      .org  0x0000
      jr    _reset

      .text

_reset:
    ld.bu   0x3fff0030[r11],r15
    jr      _reset
```

Command Sequence

ccv850e -c v030107a.850

test.o:

```
                ld.bu 0x3fff0030[r11],r15
0x600  reset:           0e4b0031          movhi   0x31, r11, r1
0x604  reset+0x4:       7f810001          ld.bu   0[r1], r15
                jr _reset
0x608  reset+0x8:       07bffff8          jr      reset (0x600)
```

Workaround

A new compiler (PATCH to 3.5.1b4), which fixes this problem, can be downloaded from the NEC WEB page(s) at http://www.renesas.eu/updates?id=443

No. a20	**Incorrect behavior on delete function optimization option "-OD"**

Version Information

V3.5.1, V4.0.5, V4.0.5c

Details

Incorrect behavior on delete function optimization option "-OD". The option "-OD" has no influence. The code generation for specified function will not be skipped. The initial code of the function is still part of the executable.

Example:

init.c:

```
void wait(void)
{
  int a=0;
  while(a<35000)
  {
    a++;
  }
}
```

main.c:

```
extern void wait(void);
void main(void)
{
  while(1)
  {
    wait();
  }
}
```

Command Sequence

ccv850e -noobj -g main.c init.c -OI=wait -OD=wait -o v030121a

Workaround

Use linker option “-delete”. This causes the linker to remove functions that are not referenced in the final executable.

No. a21	**Loop optimization on volatile access generates endless loop**

Version Information

V3.5.1

Details

Loop optimization on volatile access generates endless loop. The second query on volatile char CANES0 causes an endless loop.

Example:

test.c:

```
#define CANC0   *((volatile unsigned char  *) (0x03fec121))
#define CANES0  *((volatile unsigned char  *) (0x03fec125))

void main (void)
{
    CANC0= 0x01;                    // Disable CAN activity
    while ((CANES0 & 0x08)==0);     // Wait for INIT bit set
    CANC0= 0x00;                    // Enable CAN activity
    while ((CANES0 & 0x08));        // Wait for INIT bit clear
}
```

Command Sequence

ccv850e -noobj -g -list -passsource -Oloop v030211a.c -o v030211a

Workaround

A new compiler (PATCH to 3.5.1b4), which fixes this problem, can be downloaded from the NEC WEB page(s) at http://www.renesas.eu/updates?id=443

No. a22	**Unresolved static functions after two path inlining**

Version Information

V3.5.1

Details

Using two path inlining on static functions causes that these functions can not be resolved during linking stage. The problem is related to the amount and to the complexity of the static inlining functions.

Workaround

A new compiler (PATCH to 3.5.1b4), which fixes this problem, can be downloaded from the NEC WEB page(s) at http://www.renesas.eu/updates?id=443

No. a23	**Automatically open editor on errors**

Version Information

V3.5.1

Details

The function “automatically open editor on errors” doesn’t work using the compatibility mode compiler.

Workaround

Use option “–c2000” to compile with the next generation compiler.

No. a24	**GNUGREP generates application error**

Version Information

V3.5.1

Details

The utility “gnugrep.exe” crashes when any filename on its command Line is more than 127 characters.

Workaround

None.

No. a25	**Multiply defined type identifier on virtual functions**

Version Information

V3.5.1

Details

In a C++ project including virtual functions and section renaming the ELXR linker generates the following error message:

```
[elxr] (error) symbol type identifier for BaseClass multiply defined in
out\file1.o
out\file2.o
[elxr] (error) symbol type identifier for BaseClass multiply defined in
out\file1.o
out\file3.o
```

(repeated many times for different combinations of files, but the same symbol.)

The symbol "BaseClass" is a class containing virtual functions only. Several subclasses are derived from it. The header file “BaseClass.h” contains this declaration (reduced):

```
Class BaseClass {
  public:
              virtual int func1(int address);
              virtual int func2(int pattern);
  private:
               };
```

The problem is related to the section renaming during compile time:

```
  #pragma ghs section bss=".mybss"
```

Virtual functions sometimes require a global variable to be used as a type identifier. These global variables only occupy a single byte, but are used to associate virtual functions with the same type. Because declarations are often declared entirely in header files, the compiler has no easy way to allocate a single global variable. Instead, it tries to declare a “common” global variable in every file that needs it. These common variables are all combined at link time.
But when “#pragma ghs section bss=” is used to rename the “.bss” section, “common” variables do not longer exist. Instead, definitions are created in every file, resulting in the multiple definitions.

Workaround

1) Use section renaming during linking stage.

2) Enable the option “Force Virtual Tables” on the file “file1.cpp” and enable the option “Suppress Virtual Tables” at the top level of the project. This will cause virtual tables to be created in “file1.o” and not in any other file. With some applications this is difficult because there is no single file or group of files which contain exactly the right set of virtual tables.

No. a26	**"-no_callt" option caused wrong restore value for register R2 within ISR**

Version Information

V3.5.1

Details

By using -no_callt option, a wrong value for register R2 is restored from stack within interrupt service routine.

Example:

test.c:

```
volatile unsigned int foo;

void main(void)
{
    asm("mov 0x5a5a5a5a,r2");
    EI();
    while(1)
    {
    }
}

#pragma ghs interrupt
void timer(void)
{
    foo++;
}
```

Command Sequence

ccv850e -c -noobj -g -no_callt -no_inline_prologue v040130a.c
ccv850e -c -g startup.850
ccv850e locate.ld -entry=_reset -no_callt startup.o v040130a.o -o v040130a

Workaround

A new compiler (PATCH to 3.5.1b4), which fixes this problem, can be downloaded from the NEC WEB page(s) at http://www.renesas.eu/updates?id=443

No. a28	**Bad epilogue generation with “-prepare_dispose” and “-notda”**

Version Information

V3.5.1

Details

The compiler generates wrong epilogue sequence within interrupt service routine with options “-prepare_dispose” and “-notda”. The stack address is not reloaded into the EP after the function call. Therefore the epilogue sequence restores wrong register values.

Example:

test.c:

```
void foo(void)
{
    asm("mov 0x5a5a5a5a,ep");
}

#pragma ghs interrupt
void timer(void)
{
    foo();
}
```

Command Sequence

ccv850e –noobj -c -g -list –prepare_dispose -notda test.c

Workaround

A new compiler (PATCH to 3.5.1b4), which fixes this problem, can be downloaded from the NEC WEB page(s) at http://www.renesas.eu/updates?id=443

Version Information

V3.5.1

Details

The compiler generates incorrect accesses to a volatile object. The problem only arises when the set1 / clr1 instructions are used.

Example: test.c:

```
typedef struct Outputs_T {
    unsigned char High:1 ;
    unsigned char Low:1  ; } Outputs_T;

struct st_P3
{       unsigned int    P30_b :1;
        unsigned int    P31_b :1;
        unsigned int    P32_b :1;
        unsigned int    P33_b :1;
        unsigned int    P34_b :1;
        unsigned int    P35_b :1;
        unsigned int    P36_b :1;
        unsigned int    P37_b :1;
        unsigned int    P38_b :1;
        unsigned int    P39_b :1;};

struct st_P5
{       unsigned int    P50_b :1;
        unsigned int    P51_b :1;
        unsigned int    P52_b :1;
        unsigned int    P53_b :1;
        unsigned int    P54_b :1;
        unsigned int    P55_b :1;};

struct st_SFR
{       struct st_P3    P3;
        struct st_P5    P5;};

volatile struct st_SFR REG;

void Output(Outputs_T APOut)
{
    REG.P3.P39_b =3D APOut.High;
    REG.P5.P50_b =3D APOut.Low;
}
void main(void)
{
    Outputs_T a;
    while(1) { Output(a); }
}
```

No. a29	**Wrong optimization for "Common-subexpr elimination”**

Version Information

V3.5.1

Details

The compiler makes wrong optimization for "Common-subexpr elimination".

Example:

test.c:

```
unsigned char Buffer;

void SetBit(unsigned char Index)
{
   if  ((Index) & (0x01))
   {
      Buffer |= ((1 << ((Index - 1) >> 1)) << 4) ;
   }
   else
   {
      Buffer |= (1 << (Index >> 1)) ;
   }
}

void main(void)
{
   unsigned char i;

   for(i = 0;i < 8;i++)
   {
     SetBit(i);
     Buffer=0x00;
   }
}
```

Command Sequence

ccv850e -g -noobj -Ospeed v040316a.c -o test.out
multi -remote sim850 -cpu=V850E test.out

test.s:

```
void SetBit(unsigned char Index)
{
0x10a0  SetBit:         0086              zxb     r6
   if  ((Index) & (0x01))
0x10a2  SetBit+0x2:     86c60001          andi    0x1, r6, r16
0x10a6  SetBit+0x6:     0dd2              be      SetBit+0x20 (0x10c0)
   {
      Buffer |= ((1 << ((Index - 1) >> 1)) << 4) ;
0x10a8  SetBit+0x8:     56060001          addi    1, r6, r10

<--- wrong calculation, should be "addi -1, r6, r10"

0x10ac  SetBit+0xc:     52a1              sar     1, r10
0x10ae  SetBit+0xe:     52c4              shl     4, r10
0x10b0  SetBit+0x10:    16400030          movhi   0x30, zero, r2

<--- shift left missing

0x10b4  SetBit+0x14:    8f820105          ld.bu   0x104[r2], r17
0x10b8  SetBit+0x18:    16220104          movea   0x104, r2, r2
0x10bc  SetBit+0x1c:    890a              or      r10, r17
0x10be  SetBit+0x1e:    0dc5              br      SetBit+0x36 (0x10d6)
   }
   else
   {
      Buffer |= (1 << (Index >> 1)) ;
0x10c0  SetBit+0x20:    32a1              sar     1, r6
0x10c2  SetBit+0x22:    7201              mov     1, r14
0x10c4  SetBit+0x24:    77e600c0          shl     r6, r14
0x10c8  SetBit+0x28:    16400030          movhi   0x30, zero, r2
0x10cc  SetBit+0x2c:    8f820105          ld.bu   0x104[r2], r17
0x10d0  SetBit+0x30:    16220104          movea   0x104, r2, r2
0x10d4  SetBit+0x34:    890e              or      r14, r17
0x10d6  SetBit+0x36:    8f420000          st.b    r17, 0[r2]
   }
}
0x10da  SetBit+0x3a:    007f              jmp     [lp]
```

Workaround

A new compiler (PATCH to 3.5.1b4), which fixes this problem, can be downloaded from the NEC WEB page(s) at http://www.renesas.eu/updates?id=443

No. a30	**Wrong address calculation**

Version Information

V3.5.1

Details

The compiler makes wrong address calculation. A peephole optimization mixes up the calculation of the index variable 'channel' with the base address of two arrays.

Example:

test.c:

```
struct {
    unsigned short One ;
    unsigned short Two ;
} flagArray[5];
struct {
    volatile unsigned short *pModule[5];
} Register;

int GetStatus(int channel)
{
  int result = flagArray[channel].One;
  if (flagArray[channel].Two)
    result |= 2;
  else if ( Register.pModule[channel][0] & 0x0002)
    result |= 8;
  else if ( Register.pModule[channel][0] & 0x0800)
    result |= 4;
  GetPSW();
}
```

Command Sequence

ccv850e -passsource -Ospace test.c -S -registermode=22
-prepare_dispose

No. a30	Wrong address calculation

test.s:

```
--9: int CanGetStatus(int channel) {
             prepare    {r29,r31},0
--10:   int result = flagArray[channel].One;
             shl        2,r6
             movhi      hi(_flagArray),r6,r6
             mov        r6,r2       <--- r6 is no longer channel * 4
             ld.hu      lo(_flagArray)[r6],r29
             movea      lo(_flagArray),r6,r6
--11:   if (flagArray[channel].Two)
             ld.hu      2[r6],r10
             cmp        zero,r10
             be         .L4
--12:     result |= 2;
             ori        2,r29,r29
             br         .L6
.L4:
--13:   else if ( Register.pModule[channel][0] & 0x0002)
             movhi      hi(_Register),zero,r8
             mov        r2,r9       <--- use r2 as if it is channel * 4
             add        r8,r9
             ld.w       lo(_Register)[r9],r6
             ld.h       0[r6],r6
             shr        2,r6
             bnl        .L7
--14:     result |= 8;
             ori        8,r29,r29
             br         .L6
.L7:
--15:   else if ( Register.pModule[channel][0] & 0x0800)
             movhi      hi(_Register),r2,r2
             ld.w       lo(_Register)[r2],r6
             ld.h       0[r6],r6
             shr        12,r6
             bnl        .L6
--16:     result |= 4;
             ori        4,r29,r29
.L6:
--17:   GetPSW();
             jarl       _GetPSW,lp
--18:   return result; }
             mov        r29,r10
             dispose    0,{r29,r31},[lp]
```

Workaround

A new compiler (PATCH to 3.5.1b4), which fixes this problem, can be downloaded from the NEC WEB page(s) at http://www.renesas.eu/updates?id=443

No. a31	**Wrong memory access optimization on SFR's volatile variables .**

Version Information

V3.5.1

Details

The compiler makes wrong memory access optimization on SFR's volatile variables.

Example:
In the list file below, after execution of the 'and' instructions, r8 holds variable 'address' and r6 hold variable 'data' contents. The contents of register r6 (data) is written first, then r8 (address). As both target variables FLRAP and FLRDTL/H are volatile type, this rearrangement is wrong.

test.c:

```
#define FLRDTL_ADRS 0xfffffcc8
#define FLRDTH_ADRS 0xfffffcc9
#define FLRAP_ADRS  0xfffffcca

#define FLRDTL  *((volatile unsigned char  *) FLRDTL_ADRS)
#define FLRDTH  *((volatile unsigned char  *) FLRDTH_ADRS)
#define FLRAP   *((volatile unsigned short *) FLRAP_ADRS)

void Set_DPRAM8( unsigned short address, unsigned char data )
{
    if( (address & 0x0001) == 0x0000 )
    {
        FLRAP = address;
        FLRDTL = data;
    }
    else
    {
        FLRAP = address-1;
        FLRDTH = data;
    }
}
```

Command Sequence

ccv850e –c –noobj -Ospace test.c

No. a31	**Wrong memory access optimization on SFR's volatile variables .**

test.s:

```
void Set_DPRAM8( unsigned short address, unsigned char data )
{
0x6a0  Set_DPRAM8:            00c6              zxh     r6
0x6a2  Set_DPRAM8+0x2:        0087              zxb     r7
    if( (address & 0x0001) == 0x0000 )
0x6a4  Set_DPRAM8+0x4:        86c60001          andi    0x1, r6, r16
0x6a8  Set_DPRAM8+0x8:        6806              mov     r6, r13
0x6aa  Set_DPRAM8+0xa:        05ca              bne     Set_DPRAM8+0x12
(0x6b2)
    {
        FLRAP = address;
        FLRDTL = data;
0x6ac  Set_DPRAM8+0xc:        3f40fcc8          st.b    r7, -824[zero]
0x6b0  Set_DPRAM8+0x10:       05c5              br      Set_DPRAM8+0x18
(0x6b8)
    }
    else
    {
        FLRAP = address-1;
0x6b2  Set_DPRAM8+0x12:       6a5f              add     -1, r13
        FLRDTH = data;
0x6b4  Set_DPRAM8+0x14:       3f40fcc9          st.b    r7, -823[zero]
0x6b8  Set_DPRAM8+0x18:       6f60fcca          st.h    r13, -822[zero]
    }
}
0x6bc  Set_DPRAM8+0x1c:       007f              jmp     [lp]
```

Workaround

A new compiler (PATCH to 3.5.1b4), which fixes this problem, can be downloaded from the NEC WEB page(s) at http://www.renesas.eu/updates?id=443

No. a32	**Syntax error on a rept inside a macro .**

Version Information

V3.5.1

Details

The compiler gives a syntax error on the invocation of a macro, which invokes a repetition. The syntax error appears if the end of the rept block and the end of the macro are not immediate.

Example:
In the following test file, blank lines are inserted between endr and endm to demonstrate the problem, but it may be replaced with actual code or comments with the same effect.

test.s:

```
.macro      INCREMENT
            add 1,r6
.endm

.macro      COUNT
.rept       2
            INCREMENT
.endr
            mov r6,r10
.endm

      .text
      .org  0x0
      mov 0,r6
loop: COUNT
      jr loop
```

Command Sequence

ccv850e –c test.s

```
Error message:
test.s 20: INCREMENT
test.s 20: ^-
test.s 20: syntax error
```

No. a32	**Syntax error on a rept inside a macro.**

Workaround

Place the repetition block into a separate assembler macro.

Example:

```
.macro      INCREMENT
            add 1,r6
.endm

.macro      TEMP_COUNT
.rept       2
            INCREMENT
.endr
.endm

.macro      COUNT
            TEMP_COUNT
            mov r6,r10
.endm

      .text
      .org  0x0

      mov 0,r6
loop: COUNT
      jr loop
```

No. a33	**The use of the bit instruction NOT1 is not systematic in XOR operation**

Version Information

V3.5

Details

when compiling C code using the ^ (xor) bit operation with size optimization, the assembly instruction NOT1 is not used in all cases, although its use will reduce the code size.

Example:

test.c

```
struct bitf {
             unsigned char bit00:1;
             unsigned char bit01:1;
             unsigned char bit02:1;
             unsigned char bit03:1;
             unsigned char bit04:1;
             unsigned char bit05:1;
             unsigned char bit06:1;
             unsigned char bit07:1;
             unsigned char bit08:1;
             unsigned char bit09:1;
             unsigned char bit10:1;
             unsigned char bit11:1;
             unsigned char bit12:1;
             unsigned char bit13:1;
             unsigned char bit14:1;
             unsigned char bit15:1;
};

// bit 3 of port 5
#define      P53   ((volatile struct bitf *)0xfffff40a)->bit03
// Port 5 register
#define      P5    *((volatile unsigned char  *)0xfffff40a)

void main(void)
{
    P5 ^= 0x08;
    P53 ^= 1;
}
```

Command Sequence

ccv850e -OS -S test.c

No. a33	The use of the bit instruction NOT1 is not systematic in XOR operation

Generated assembly code:

```
void main(void)
{
    P5 ^= 0x08;
0x1094  main:           8e20f40a   movea   0xfffff40a, zero, r17
0x1098  main+0x4:       5fd10000   not1    3, 0[r17]
    P53 ^= 1;
0x109c  main+0x8:       8700f40a   ld.b    -3062[zero], r16
0x10a0  main+0xc:       86b00008   xori    0x8, r16, r16
0x10a4  main+0x10:      8740f40a   st.b    r16, -3062[zero]
0x10a8  main+0x14:      7f00f40a   ld.b    -3062[zero], r15
}
0x10ac  main+0x18:      007f   jmp     [lp]
```

Workaround

The user should not use a 1-bit variable.

No. a34	***Debug command print fails if a variable's type has the same name as one of the command option***

Version Information

V3.5

Details

With the debug command line print, the user gets a parse error when in the code a typedef has the same name as the option passed to print.

Example:
test.c

```
typedef unsigned char c;

c dummy;
int main()
{
            dummy = 2;
            return dummy;
}
```

Command Sequence

In debugger command pane:

```
MULTI> print /c dummy
parse error
```

Workaround

The print command has the following syntax: print[/format] exp
With format: [count] style[size]

As soon as the option 'count' or 'size' is used then the problem disappears:
MULTI>print /1b toto
toto =...

Or change the name of the typedef.

No. a35	**Compiler option –nomacro doesn't work for all CPUs**

Version Information

V3.5

Details

The compiler option –nomacro is ignored with the option -cpu=?

Example:

test.s

```
------------------------------------------------------------------------
--  program start
--
------------------------------------------------------------------------

.text

.globl  _main

_main:
        mov     12,r7

        mov     16,r7
        movea   16,r0,r7

        mov     32767,r7
        movea   32767,r0,r7

        mov     0x8000,r7
        movhi   hi(0x8000),r0,r7

        mov     0x888000,r7
        movhi   hi(0x888000),r0,r7

        mov     0x88888888,r7
        movhi   hi(0x88888888),r0,r7
        movea   lo(0x88888888),r1,r7

        jmp     [lp]
```

Command Line

ccv850e -nomacro -cpu=v850e test.s

Workaround

None.

No. a36	EP is not restored in ISR that calls func_ptr

Version Information

V3.5.1

Details

If an interrupt service routine calls a function from a table, the EP register is not restored after the call.

Example:

test.c

```
#include <V850\df3231.h> /* V850 ES/FE2 register definitions */

#define     _P30        ((volatile struct bitf *)0xfffff406)->bit00
/* external data */

void pinset (void)
{
    _P30^=1;
    asm("mov 0xffffc000, ep");
}

void (*table[2])(void)= {pinset, pinset};
void main(void)
{

      while (1)
      {

      };
}
```

Interrupt.c

```
extern void(* const table[])(void);
/* Timer M equ 0 interrupt */
__interrupt void vTimerM(void)
{
    table[0]();
}
```

Command Sequence

ccv850e –prepare_dispose test.c interrupt.c

No. a36	EP is not restored in ISR that calls func_ptr

Workaround

1. Compile with –no_prepare_dispose

2. Call a dummy function directly in the interrupt function. NOTE: this function may be optimised away when the option -Ospace is used.

```
extern void (*const table[])(void);

void  dummy(void) {}

void interrupt()
{
    dummy();
    table[0]();
}
```

3. Compile with the option –tda and put a dummy assignment to a TDA variable in the interrupt function.

```
extern void (*const table[])(void);

#pragma ghs starttda
char dummy;
#pragma ghs endtda

#pragma ghs interrupt
void interrupt()
{
    dummy = 0;
    table[0]();
}
```

No. a37	**The debug command copy has a size limit**

Version Information

V3.5

Details

The Debug command “copy” cannot copy more than 0x9999 bytes. Trying to copy more doesn’t bring any error message.

Command Sequence

```
MULTI> target fill 0x0 0x10000 0x88 1
MULTI> copy 0x0 0x40000 0x10000 1
MULTI> target m 0x04fff0 l=0x12
0x0004FFF0: 8888 8888 8888 8888 8888 8888 8888 0088
0x00050000: 0000
```

Workaround

The user can use the target command copy or use several time the debug command copy.

No. a38	**tda-offset out of range in sld.bu instruction**

Version Information

V3.5, V4.0.5, V4.0.5c

Details

The instruction sld.bu from the V850ES core accepts only a 4-bit offset.
When using tda, the compiler generates assembler code using the sld.bu instruction with a bigger offset and an assembler or linker error is given.

Command Line

ccv850e –cpu=v850e main.c –noobj (Assembler error)
ccv850e –cpu=v850e main.c –obj (Linker error)

Workaround

1. Do not use tiny data area (-notda)
2. Use another CPU that does not have this instruction.

No. a39	***In case of redundant load, an unsigned load is considered unnecessary in case a signed load to the same location was previously done.***

Version Information

V4.0.5

Details

When a signed load is already done on a variable, the compiler considers unnecessary to do an unsigned load:

Example:
test.c

```
int sub(unsigned char *res)
{
    int i = 0x80, j = 0;
    while (i != 0)
    {
      if(1 & *res)
          j = i;
        i >>= 1;
        *res >>= 1;
    }
   return j;
}
```

No. a39	**In case of redundant load, an unsigned load is considered unnecessary in case a signed load to the same location was previously done.**

ASM- Code:

```
_sub:
      movea 128,zero,r2
      mov   0,r10
.L6:
--    if(1 & *res) ld.b 0[r6],r17   -- signed load
      mov   r17,r16
      shr   1,r16
      bnl   .L7
      mov   r2,r10
.L7:
      sar   1,r2
--    *res >>= 1; shr   1,r17 -- doesn't reload the value as unsigned.
      st.b  r17,0[r6]
      cmp   zero,r2
      bne   .L6
```

Command Sequence

ccv850e –Ospace test.c
ccv850e –Ospace

Workaround

A new compiler Ver. 4.05c, which fixes this problem, can be downloaded from the NEC WEB page(s) at http://www.renesas.eu/updates?id=443

No. a40	**In the expression, int a %=(1+~(unsigned b)), only the % is done.**

Version Information

V3.5, V4.0.5

Details

Wrong code is generated for int a %= (1 + ~(unsigned b))

Example:
test.c

```
void sub(unsigned b)
{
int a = 29970;a%= (1 + ~b);}
```

Command Sequence

ccv850 –Ospace
ccv850e –Ospace

Generated ASM-Code:

```
-- int a = 29970;
movea 29970,zero,r17
-- a%= (1 + ~b);
divu  r6,r17,r2
```

Workaround

A new compiler Version 4.05c, which fixes this problem, can be downloaded from the NEC WEB page(s) at http://www.renesas.eu/updates?id=443

No. a41	**Assignment to a bit field element gives an Internal Compiler Error (ICE) when compiled with the option –Ospace**

Version Information

V4.0.5

Details

When compiling with the option –Ospace, if an array of bit fields with a size equal to 8 or 16 is used, an internal compiler error occurs when an array element is assigned to another array element.

Example:
test.c

```
void func(void)
{
    struct s
    {
      int bit:8;
    }data[] = {0,0};

    data[0].bit |=data[1].bit;      <- this line gives the ICE
}
```

Command Sequence

ccv850 –Ospace
ccv850e –Ospace

Workaround

A new compiler Version 4.05c, which fixes this problem, can be downloaded from the NEC WEB page(s) at http://www.renesas.eu/updates?id=443

No. a42	When compiled with –Ospace, an assignment to a structure's element will also set to 0 all other elements

Version Information

V4.0.5

Details

When compiling with the option –Ospace, if anelement of structure variable is assigned to another structure element, the compiler will set to 0 the other elements of the base structure.

Example:
test.c

```
struct  data
{
    int m0:2;
    int m1:2;
} s[2], *p=s;

void sub(void)
{
    p->m0 = p[1].m0 = 1;
    p->m1 = p[1].m1 = 1;

    p->m1 &= p[1].m1;          ← Wrong code generated.
}
p->m0 and all other fields of the structure will be set to 0.
```

Command Sequence

ccv850 -Ospace
ccv850e -Ospace

Workaround

A new compiler Version 4.05c, which fixes this problem, can be downloaded from the NEC WEB page(s) at http://www.renesas.eu/updates?id=443

No. a43	**In the expression, n=s.m++ & s.m++, the incrementation is done only once.**

Version Information

V4.0.5

Details

The compiler generates wrong code for n=s.m++ & s.m++. The incrementation is done only once after the & operation..

Example:
test.c

```
struct data
{
    int m1;
} str;

int main(void)
{
    int n = str.m1++ & str.m1++;
    return n;
}
```

Command Sequence

```
ccv850 -Ospace
ccv850e -Ospace
```

```
Generated Code:
_main:
      movhi hi(_str),zero,r11
      movea lo(_str),r11,ep
      sld.w 0[ep],r2
      mov   r2,r10
      and   r2,r10
      add   1,r2
      sst.w r2,0[ep]
      jmp   [lp]
```

Workaround

A new compiler Version 4.05c, which fixes this problem, can be downloaded from the NEC WEB page(s) at http://www.renesas.eu/updates?id=443

No. a44	**When compiling with –Ospace, the operator | is replaced with ||. This implies that the second expression is skipped if the first one is true.**

Version Information

V4.0.5

Details

When compiling with –Ospace, it may happen that the operator | is replaced with ||. This implies that the second expression '(2!=b++)' is skipped if the first one is true and this even if the second member has side effect on the rest of the code. The exact condition is shown below. The condition is not met, if the first expression is hold simple and does not contain an operation with Boolean expressions.

Example:
test.c

```
int main(void)
{
    int a =1;
    int b = 0;
    a = (a !=0) | (2 != b++) ? 10:20;
    return b;
}     // In this example, the increment 'b++' is not done.
```

Command Sequence

ccv850 -Ospace
ccv850e -Ospace

Workaround

A new compiler Version 4.05c, which fixes this problem, can be downloaded from the NEC WEB page(s) at http://www.renesas.eu/updates?id=443

No. a45	**The environment affects the code generated.**

Version Information

V3.5

Details

The directory path changed the behavior of the compiler. Correct but different code is generated depending on the project location.

Workaround

Use compiler version 4.07.

No. a46	**When connected to an emulator, the response time of the debugger is long when the I/O window is waiting for an input.**

Version Information

V4.0.5

Details

When connected to an emulator, the debugger (v4.0.5) response time is very slow in the following case:

> *The input/output window is waiting for an input because of a "getchar()" or a "scanf()" in the code.*

Switching between debugger command pane, target window or openning the data explorer is taking minutes and sometimes if more than one request is done then the debugger just hangs.

A solution is to disconnect from the target from the builder or the launcher window

Workaround

None

Recommendations:

1. Switch to the I/O window before hitting the *"getchar()"* or *"scanf()*".
2. Use latest 850eserv available on http://www.renesas.eu/updates?id=443

No. a47	**The linker option –codefactor breaks the code in interrupt functions when code is optimized for speed**

Version Information

V4.0.5, V4.0.5c

Details

The linker option -codefactor modifies the prologue of an interrupt routine in a way that writes on r6 and r7. This is correct in non-interrupt routines but not here, overwriting those registers might cause the code to break

Command Line

ccv850e –codefactor – Ospeed

Workaround

Use new compiler Version 4.07.

No. a48	**Any macro containing a sequence with a '.' followed by other characters is not expanded correctly.**

Version Information

V3.5, V4.0.5, V4.0.5c

Details

When a macro like this:

```
#define LEAF( name) .section name;
```

is expanded by the preprocessor, the result is a white space separated text like this

```
. section name
```

Workaround

1. Use option "`-option=98`" to preprocess as expected.
2. Use new compiler Version 4.07.

No. a49	**Short Load instruction patch failure**

Version Information

V4.0.5c

Details

This effect is based on above described ‘a38’ related to sld.bu access. The compiler version 4.05c generates a correct access now, but linker introduces a ‘jarl’ instruction instead of ‘sld.bu’.

Command Line

ccv850e –cpu=v850e main.c

Workaround

1. Do not use tiny data area (-notda)
2. Use another CPU that does not have this instruction.
3. Use new compiler Version 4.07.

No. a50	r20/r21 used by codefactor incorrectly

Version Information

V4.0.5, V4.0.5c

Details

When using the linker optimization '-codefactor' with mask register options, the mask register might be overwritten with different value.

Command Line

ccv850e –cpu=v850e -Ospace --inline_tiny_function –codefactor -r20has255. main.c

Workaround

1. Do not use '-codefactor' linker optimization along with '–r20has255' and '-Ospace' within Multi2000 Builder.
2. If linker (ELXR) is invoked on command line with option '-r20has255' directly passed, the generated code is correct.
3. Use new compiler Version 4.07.

No. a51	The linker give an application error when the option –gnu_linkmap is passed to the compiler driver

Version Information

V4.0.5, V4.0.5c

Details

When passing the linker option –gnu_linkmap to the compiler driver and linking more than one object file, the linker generates an application error.

Command Line

ccv850e –cpu=v850e locate.ld –gnu_linkmap=test.txt main.o ints.o startup.o –o project.out

Workaround

Download the new linker form our website http://www.renesas.eu/updates?id=443

No. a52	V850e1f (Floating point unit): Wrong code generation for floating point comparisons

Version Information

V3.5, V4.0.5, V4.0.5c,V4.0.7

Details

When the CPU V850e1f (Floating Point Unit) is selected, the compiler generates code using the floating point instructions SUBF, ADDF, MULF and DIVF, which do not affect the OV-flag. But the compiler is using the OVERFLOW flag 'OV' for the evaluation of the result.

```
int fcmp( float fSat)
{
    if( fSat < 0.0f)
       return -1;
    return 0;
}

if( fSat < 0.0f)
//        mov           0,r17
//        subf.s        r17,r6,zero
//        trff
//        movea         0,r1,r1
//        bge           .L4        <== should be 'BL'-instruction
return -1;
//        mov           -1,r2
//        br            .L2
...
```

Command Line

ccv850e -cpu=v850e1f -c -list -passsource test.c

Workaround

Download a compiler update from the internet http://www.renesas.eu/updates?id=443

This patch will update all compiler versions to new 4.07A. The patch does not change the GUI portions of the compiler, but underlying compilers and libraries.

No. a53	Single Step Limitation using OCD N-Wire Debugger

Version Information

All Versions

Details

A single step over a switch statement may fail, since the debugger may require activating more SW breakpoints than available with N-WIRE debugging method.

In this case an Error message is displayed:

“Couldn't set breakpoint at 0xb38
No more ROM area software breakpoints available
warning: Could not set breakpoint, disabling.”

Command Line

None

Workaround

Evaluate the switch parameter and set a break point at the target case.

No. a54	**Peephole Optimizer does not generate proper code**

Version Information

V4.0.5, V4.0.5c, V4.0.7

Details

The peephole optimizer is not recognizing various items regarding redundant code recognition and internal temporary register references. Although the generated code is not wrong, the optimisation is not successfully done.

Command Line

ccv850 –c –Ospace –list –passsource test.c

The compiler generated a sequence:

```
--1193: static void copy_block(s, buf, len, header)
      .align      2
      .globl ___copy_block_static_in_trees_inf
___copy_block_static_in_trees_inf:
      prepare     {r28,r29},0
      mov   r6,r2
      …
      dispose     0,{r28,r29},[lp]
```

Since the prepare/dispose header are unnecessary, if only temporary registers are used, they can be replaced by

```
      jmp   [lp]
```

Workaround

Download a compiler update from the internet at

http://www.renesas.eu/updates?id=443

This patch will update all compiler versions to new 4.07A. The patch does not change the GUI portions of the compiler, but underlying compilers and libraries.

No. a55	**Option '-passsource' may affect code generation**

Version Information

V4.0.5, V4.0.5c, V4.0.7

Details

The option '-passsource' may lead to different code generation compared to modules not compiled with it.

Command Line

ccv850 –c –list –passsource test.c

Workaround

Download a compiler update from the internet at

http://www.renesas.eu/updates?id=443

This patch will update all compiler versions to new 4.07A. The patch does not change the GUI portions of the compiler, but underlying compilers and libraries.

No. a56	**The linker creates only one far_call patch function per label**

Version Information

V3.5.1, V4.0.5, V4.0.5c, V4.0.7, V4.0.7a

Details

The linker creates only one far_call patch function per label, if the label is addressed in the following way:

```
.extern
__NMI0:
  jr _vectable+0x8
  .offset        0x20
__NMI1:
  jr _vectable+0x4
```

The linker is taking the first reference to vectable + offset and uses this address to generate the farcall patch function. It reuses it for all references to vectable. The consequence is that the generated code is wrong, causing the program to fail.

Command Line

Ccv850 -c

Workaround

1. Use only labels, which can be reached without far_callpatch() insertion of the linker, or
2. Use latest Patch, latest tool version, or
3. Use only unique labels like this:

```
__NMI0:
  jr _vectable_08
  .offset        0x20
__NMI1:
  jr _vectable_04
```

No. a57	**The Multi Debugger displays large volatile structures very slowly**

Version Information

V4.0.5, V4.0.5c, V4.0.7a

Details

If a huge structure pointer with memory attribute ‘volatile’ is displayed, the update speed of the debugger slows down dramatically. For example a structure with approx. 7000 Bytes requires 40 seconds for an update.

Command Line

None.

Workaround

1. View always only parts of the structure, not the entire object.
2. Do not use ‘volatile’ keyword, if possible.

No. a58	**Compiler seg-fault with –farcalls –pic –mtda**

Version Information

V3.5.1, V3.5.1b3, V3.5.1b4

Details

When compiling the following code, the compiler give an internal error:
Internal Program Error (11)

Test2.c

```
extern void func(void)
void test(void)
{
  func();
}
```

Command Line

ccv850e –S –farcalls –pic –mtda test2.c

Workaround

None

No. a59	**Incorrect conditional branch after tst1**

Version Information

V3.5.1, V3.5.1b3, V3.5.1b4

Details

In the example below the compiler uses the wrong branch instruction after a bit test with the tst1 instruction:

```
int byte, word;

int TestFunk(void)
{
   if ((((byte & 1) == 1) == 1) && word)
            movhi hi(_byte),zero,r1
            tst1  0,lo(_byte)[r1]
            setfne      r2
            bne   .L4
            movhi hi(_word),zero,r16
            ld.w  lo(_word)[r16],r16
            cmp   zero,r16
            be    .L4
      return 5;
            mov   5,r10
            br    .L2
      .L4:
   else if (((byte & 1) == 1) == 1)
            cmp   1,r2
            setfne      r10
      .L2:
      return 0;
   return 1;
      jmp   [lp]
}
```

Command Line

ccv850e -cpu=v850e -Ospace

Workaround

1. The common sub-expression elimination should be disabled (-Onocse)
2. A new compiler update (3.5.1B5) is available for download.

No. a60	**A double constant might be converted to long long.**

Version Information

V3.5.1, V3.5.1b3, V3.5.1b4

Details

The floating point const of the following example is converted to a long long at compile time. The results of the operation are in consequence not valid.

Example:

```
------------
const double LSB=10.0E-12;

long long f2(void)
{
  return 100.0/LSB
          mov     -1, r12
          mov     0, r13
}
          mov     r12, r10
          mov     r13, r11
          jmp     [lp]
```

Command Line

ccv850e –cpu=v850e

Workaround

Use latest patch available under http:/www.renesas.eu/updates?id=443

No. a61	**Far call patch of interrupt prologue function overwrites R1**

Version Information

V3.5.1, V4.0.5

Details

When compiling with –Ospace, the GHS compiler uses its library prologue and epilogue functions.
If the function is too far away for jarl instruction, the compiler introduces a far call patch function.
This is not a problem in any other functions but interrupt service routine.
Indeed the register R1 will be destroyed before being saved.

Example:

```
------------
#pragma ghs interrupt
void INTTM0EQ0(void) //timer valid edge / match
        addi    0xffffffa0, sp, sp
        st.w    lp, 0x5c[sp]
        jarl    _ghscallpatch___ghs32rsave00i_0 (0x7da), lp

_ghscallpatch___ghs32rsave00i_0:
        mov     0x10008c8, r1
        jmp     [r1]
```

Command Line

ccv850e –Ospace

Workaround

1. Use –prepare_dispose
2. Use –inline_prologue

No. a62	**16-bit multiply and divide are used instead of 32-bit operations**

Version Information

V3.5.1

Details

A 16-bit multiply/divide operation is in some cases used when a 32-bit operation is needed. This is due to the peephole optimisation which doesn't check the type of size of operation parameter to determine the legality of the operation.

Example:

```
------------
void sub2(unsigned short);

void ng_new(short para1,short para2)
        add     -4,sp
        st.w    lp,0[sp]
{
    unsigned short  data;
    if(para1<123)
    {
        sxh     r7
        sxh     r6
        addi    -123,r6,zero
        bge     .L4
      data = (unsigned short)((para2 * (240 - para1) * 17) >> 12) >>10;
        movea   240,zero,r15
        sub     r6,r15
        mul     r15,r7,zero
        mulhi   17,r7,r11                   <- this should 32bit x 32bit
        shr     12,r11
        andi    64512,r11,r11
        sar     10,r11
        mov     r11,r6
        zxh     r6
.L4:
    }
    sub2(data);
        jarl    _sub2,lp
```

Command Line

ccv850e –O
ccv850e –Ospace
ccv850e –Ospeed

Workaround

Disable peephole optimisation : -Onopeep

No. a63	**Linker does not delete unused functions**

Version Information

V3.5.1, V4.0.x

Details
When compiling with “–delete” and “-g” plus “-dual_debug”, the linker will not delete unused functions.

Command Line

ccv850e –delete –g –dual_debug test.o –o test.out

Workaround

A new linker available on NEC’s web site

http://www.renesas.eu/updates?id=443

Patchfile CPDW9XNT-CDR-V85X-V407A-PATCH03.zip. Use with new flag “-ignore_debug_references”.

No. a64	**Utility gsrec, ghexfile fails padding, if ROM access is split to even/odd**

Version Information

V3.5.1, V4.0.x

Details
The padding option will overwrite the contents of the other bank.

Command Line

gsrec test -bytes 16 -pad1 0 0x3FFF 0xA5 -start 0 -end 0x3FFF -romaddr 0 -interval 8:4 -o test_even.rec

ghexfile test -bytes 16 -pad1 0 0x3FFF 0xff -h -start 0 -end 0x3FFB -romaddr 0 -interval 8:4 -o test_even.hex

Workaround

Use new compiler/ linker update available on NEC’s web site
http://www.renesas.eu/updates?id=443

Patchfile CPDW9XNT-CDR-V85X-V407A-PATCH03.zip or CPDW9XNT-CDR-V85X-V351-PATCH02.zip or later

No. a65	**Make Dependency Generator creates too much dependencies**

Version Information

V3.5.1, V4.0.x

Details

When using –make, the dependency of each source file is generated by the compiler automatically, but with this option also other source files are taken into account and not only header files.

Command Line

ccv850 –make test.c

Workaround

Use new compiler Ver. 4.xx with option –MD instead of –make.

No. a66	**The compiler dereferences the wrong register in certain places due to an incorrect optimization.**

Version Information

V3.5.1

Details:

Under some conditions the compiler dereferences the wrong register causing the code to break. In the example below the register R2 was dereferenced.

```
_test:
        prepare     {r26,r27,r28,r29,r31},0

--171: if(((FALSE == Var1_bt)&&(FALSE != Var2_ft))||(FALSE != Var3_bt))

            movhi   hi(_Var3_bt),zero,r2
            movea   lo(_Var3_bt),r2,r29
            add     5,r29
            movea   sdaoff(_Var1_bt),gp,r19
            tst1    0,0[r19]
            bne     .L7
            tst1    0,0[r2]         //← error should be using -5[r29]
            bne     .L6
           /*.....*/
```

Command Line

ccv850 -c -cpu=v850e -Ospace test.c

Workaround

A patch for the V3.5.1 is available on Renesas website:

http://www.renesas.eu/updates?id=443

No. a67	**The compiler generates different object code for identical source modules.**

Version Information

V4.0.x

Details:
The compiler generates a different code for the identical source modules. This behaviour involves an enum defined inside a structure. The generated code is not wrong but the algorithm using these structures and ‘enum’s is sometimes different.

Test.c:

```
struct {
  enum { a, b, } e;
} g;

void fun1(char);

void test (int j)
{
   fun1(j);
}
```

Command Line
ccv850 -c -cpu=v850e -Ospace test.c

Workaround

Do not define enumerations inside a structure.

No. a68	**The FPU library for V850E1F contains “callt” even if omitted by the compiler**

Version Information

V3.5.1,V4.0.5, V4.0.7, V4.0.7A, V4.2.3

Details:
The floating point library for the V850E1F target CPU cores contains ‘callt’ instructions, even if they are explicitly disabled by the user.

Command Line
ccv850 -cpu=v850e1f -no_callt …

Workaround

A patch is available on

http://www.renesas.eu/updates?id=443

No. a69	**Out-Of-Order Code Generation**

Version Information

V3.5.1,V4.0.5, V4.0.7, V4.0.7A

Details:

The usage of an assembler macro within a preprocessor defintion decaolaring a new block may result in out-of-order code generation, if used within a function and the structure packing feature is used.

Example:

```
asm int IRQ_suspend(void) {
        stsr    PSW, r10
        di
}

asm void IRQ_restore (int oldPSW) {
%reg oldPSW
        ldsr        oldPSW, PSW
%error
}

#define ABS      \
{   // new block decalred here       \
    int psw = IRQ_suspend();         // psw is locally available only!

#define ABE      \
    IRQ_restore(__psw);          \
}           // new block ends here

void foo( void)
{
        ABS
        ...
        foosub();
        ...
        ABE
        ...
}
```

It is most likely that with –pack=1 the block end substitution code is executed directly after the block started.

Command Line

ccv850 –pack=1 test.c

Workaround

1. Use latest compiler release 4.2.3

2. For compilers Ver. 3.5.1
2.1 Do not use one of these switches
'#pragma pack(n)' or
-pack=n via commandline
if n!=8 (the default setting).

2.2 Do not create a new block within the preprocessor macro, instead use a common variable.
Example:

```
extern unsigned int __psw;   // global or static variable __psw
asm int IRQ_suspend(void) {
    stsr    PSW, r10
    di
}

asm void IRQ_restore (int oldPSW) {
%reg oldPSW
    ldsr       oldPSW, PSW
%error
}

#define ATOMIC_BLOCK_START      \
   __psw = IRQ_suspend();

#define ATOMIC_BLOCK_END        \
   IRQ_restore(__psw);          \
```

3. For compiler versions 4.0.0 up to 4.2.1
3.1 Use new intrinsic function __GETSR()/__SETSR() to handle PSW values

```
#define ATOMIC_BLOCK_START      \
{                               \
   int psw = __GETSR();         \
     __DI();

#define ATOMIC_BLOCK_END        \
   __SETSR(psw);                \
}
```

3.2 Use command line option '-misalign_pack'

No. a70	**Incorrect Warning with '#pragma ghs io'**

Version Information

V4.2.3

Details:

The compiler sometimes generates a warning message, if the the "#pragma ghs io" construct is used:

```
".\PF\MC_HAC\HAC_V850\v850_tab.h", line 1118: warning #1514-D:*
incorrect GHS #pragma: a single #pragma ghs io directive has induced
multiple I/O variables*
```

Sample Code:

```
typedef union
{
  uint8 All_ui8;
  struct
  {
    // Chip select address range
    // DCS1 DCS0  Chip select address range
    //  0     0    CS0 area (01F 8000H - 01F FFFFH)
    //  0     1    CS1 area (03F 8000H - 03F FFFFH)
    //  1     0    CS2 area (07F 8000H - 07F FFFFH)
    //  1     1    CS3 area (0FF 8000H - 0FF FFFFH) (default)
    uint8 DCS_bf2   : 2;
    uint8 res_bf5   : 5;
    // Read access control of data flash: 0: Disable 1: Enable
    uint8 DFLEN_bf1 : 1;
  } Bit_st;
} gType_DFLCTL_un;

#pragma ghs io g_DFLCTL_un 0xFFFFFCF8
volatile gType_DFLCTL_un g_DFLCTL_un;
```

Command Line

ccv850 -cpu=v850e1 –c test.c

Workaround

A patch "CPDW9XNT-CDR-V85X-V423-PATCH02" is available on
http://www.renesas.eu/updates?id=443

No. a71	**Spaces not recognized by elxr in @files**

Version Information

V4.2.3

Details:
When the linker (elxr.exe) is called directly and when all linker options are read from a file (@yyy.txt), the linker will give an "unkown symbol error XXX" if the option -e XXX is used and is not the last option of the file.

Command Line
elxr.exe @yyy.txt

Workaround

A patch “CPDW9XNT-CDR-V85X-V423-PATCH02” is available on
http://www.renesas.eu/updates?id=443

No. a72	**Comments not recognized by elxr in @files**

Version Information

V4.2.3

Details:
The linker cannot process the option file correctly which includes the line beginning with '#'. Such line should be assumed as comment but elxr tries to find the file its name begins with '#':

objlist.txt:

```
t2.o
#t3.o
t4.o
```

Command Line
ccv850e t1.c -lnk=@objlist.txt
[elxr] (error) cannot find file #t3.o

Workaround

A patch “CPDW9XNT-CDR-V85X-V423-PATCH02” is available on
http://www.renesas.eu/updates?id=443

No. a73	**Wrong Default Library Selection**

Version Information

V4.2.3

Details:
The compiler may not select the correct library when code is generated with the options
-no_callt –cpu=v850e

Although te compiler may have an option to generate code for the V850E/ES derivative, the option –no_callt advises the compiler to select the non-E type libraries. The compiler does this, but the driver decides nevertheless to select the v850e libs. This leads finally to failure in saving/restoring register r2 in an interrupt epilogue.

Command Line
ccv850e –cpu=v850e –no_callt

Workaround

Use for modules containing interrupt services the options
-prepare_dispose
-inline_prologue

This may increase the code size of the ISR slighltly, but improves interrupt response times.

No. a74	**Wrong interpretation of option '-T'**

Version Information

V4.07a, V4.2.3

Details:
If in a command line the option –T linkfile.ld is specified, it will be parsed as such, but an additional empty '-T' option is issued to the linker. This additional option is causing the linker to ask for the correct linker directive file name.

Command Line
ccv850e -T df3318x.ld -o test.out @linker.cmd

Workaround

Simply call the linker directive file directly without a preceeding '-T' option.

Example: ccv850e df3318y.ld -o test.out @linker.cmd

No. a75	**Internal Error with '-OI'**

Version Information

V4.0.7, V4.0.7a, V4.2.3

Details:
If in a command line the option '-OI' is specified, the compiler will generate an internal error, if the inlined function contains assembler instructions.

Command Line
ccv850e –c -OI -o test.out test.c

Workaround

1. Use either one of the options below to get rid of the internal error:

-list
or
-noobj

2. Use latest patch CPDW9XNT-CDR-V85X-V424-PATCH01

No. a76	**Prevent use of set1 on volatile accesses other than char**

Version Information

V4.0.5, V4.0.7, V4.0.7a, V4.2.3, V4.2.4

Details:
The use of set1/clr1 instructions is not always restricted to variables and bitfield other than size of char.

Command Line
ccv850e –c -Ospace -o test.out test.c

Workaround

Use latest patch CPDW9XNT-CDR-V85X-V424-PATCH01

No. a77	**Volatile Initializer forgotten**

Version Information

V4.0.5, V4.0.7, V4.0.7a, V4.2.3

Details:
The compiler may 'forget' to initialize structure members, if both of the compiler options "–pack=1" and "-misalign_pack" are used.

Example:

```
typedef volatile struct
{
    volatile unsigned short X;
} mytype;
unsigned char buffer[256];

void main(void)
        .text
        .align    2
        .globl    _main
_main:
{
    mytype *foo;
    foo = (mytype *)buffer;
        movea  sdaoff(_buffer),gp,r2
    foo->X = 1234;
    foo->X = 5678;
        movea  5678,zero,r10
        st.h   r10,0[r2]
…
```

Command Line
Ccv850e –cpu=v850e –Ospace -pack=1 -misalign_pack

Workaround

1. Use latest compiler CPDW9XNT-CDR-V85X-V424 or
2. ptach update CPDW9XNT-CDR-V85X-V424-PATCH01 available at
http://www.renesas.eu/updates?id=443

No. a78	**The compiler does not accept ‚float' –parameters for assembler macros**

Version Information

V4.0.5, V4.0.7, V4.0.7a, V4.2.3, V4.2.4

Details:
The compiler does not pass variable of type float to assembler macros. Instead, the parameters are converted into double types before passed.

Example:
```
asm fabsfA(x)
{
%reg x
      absf.s       x,r10
%error
}
…
X= Y* absfA( f);
```

Instead that the compiler generates an inline call of absfA, it first converst the parameter from float to double.

Command Line
ccv850e –c –cpu=v850e1f -Ospace test.c

Workaround

The status of the parameters cannot be changed, as per status of today, but for example, an inlined version of the absf() function and a sort of floor functionality can be emitted using this instructions:

```
#define fabsfA(x) (((x) < 0) ? -(x): (x))      // emmitts assembler code absf.s
#define floorfA(x) ((float)(int)(x))           // emmitts instructions trnc.sw; cvt.ws
```

No. a79	**The compiler emitts the instruction “signed load”, instead of “unsigned load”**

Version Information

V4.0.5, V4.0.7, V4.0.7a, V4.2.3

Details:
The compiler emitts a signed load of 8 or 16-bit ‘unsigned’ variables, where it should emitt only unsigned loads. This issue can be observed in comparisons where an unsigned field of size 8 or 16 is used in a comparison. A ‘field’ is here meant to be a member of a sructure or class, not a bitfield or a standalone variable. Also arrays are not affected by this.

Example:

```
int main( void ) {
   struct { unsigned char uc ; } s = { 199 } ;

   return ( s.uc != 199 );
}
```

Command Line
ccv850e –c –cpu=v850e -Ospace test.c

Workaround

V4.2.3 users can pass an option “-X3016”, or please use latest release V4.2.4 or later.

No. a80	**Wrong access order to 'volatile' arrays**

Version Information

V4.0.5, V4.0.7, V4.0.7a, V4.2.3, V4.2.4

Details:
The compiler 'forgets' the volatile attribute of certain objects. The main reason for this is the pipeline scheduler as the primary source responsible for changing the order of instructions.

Example:

```
volatile unsigned char data[20];

void testfunc(void)
{
      data[2] &= ~0x20;
      data[3] &= ~0x20;

      data[10] |= 0x20;
      data[11] |= 0x20;

      data[10] |= 0x40;
      data[11] |= 0x40;
}
```

The generated order of accessing the array is different to that specified in the source module.

```
...
--5:  data[2] &= ~0x20;
      mov   _data,r2
      set1  5,11[r2]
--11: data[10] |= 0x40;
      set1  6,11[r2]
      ld.bu 2[r2],r17
      set1  5,10[r2]
...
```

Command Line
ccv850e –c –cpu=v850e -Ospace test.c

Workaround

1) Please use newer compilers V4.2.4-N1 or later.
2) Use option –Onopipeline, if compiler update is not possible

No. a81	**Missing Compiled Line**

Version Information

V4.0.5, V4.0.7, V4.0.7a, V4.2.3, V4.2.4

Details:

To reproduce the bug, there must be the following series of instructions

```
<instruction which puts a value into X>
AND X, <constant with only 1 bit set>
<instruction which uses the condition code of the AND>
...
cmp   X,0
```

The problem is that the AND is deleted, leaving nothing to set the condition codes for the instruction which uses the condition.

Example:

```
void test(volatile char *p) {
    unsigned char temp;
    temp = (p[0] & 2);
    p[1] = (temp == 0);
    p[2] = (temp != 0);
}

#pragma pack(1)
struct { int i; } s, *p = &s;
#pragma pack()
```

Command Line

ccv850e -w -S -Ospace test.c

Workaround

1) Please use option -Z1276 for all V4.x.x compilers.
The problem is fixed in actual V5.1.6C release.

No. a82	**Compiler Option**

Version Information

V5.1.6C

Details:
The compiler accepts two new options:
***(1) -v850e_mul_errata**:*
The compiler will not emit instructions of the form

mul reg1, reg2, reg3 or mulu reg1, reg2, reg3 or
mul imm1, reg2, reg3 or mulu imm1, reg2, reg3

where reg3 is r0, or reg2 is the same as reg3

The assembler will flag any occurrences of the above instructions, but will not attempt to correct the problem.

(2) -no_v850e_mul_errata:
Can be used to disable -v850e_mul_errata

Example:
None.

Command Line
ccv850 –cpu=v850e -v850e_mul_errata test.c

Workaround
There is a patch update available CPDW9XNT-CDR-V85X-V516-PATCH03 including assembler, compiler and new libraries built with this new option, and the library sources have been audited to remove any instructions violating the above rule. The patch is available at

http://www.renesas.eu/updates?id=26

Please select the actual patch update:
CPDW9XNT-CDR-V85X-V516-PATCH03.zip

No. a83	**Support Of Bigger Trace using IECUBE2**

Version Information

V5.1.6C

Details:
The extended trace board of the IECUBE2 is now supported under GHS. This requires updates of both, the target debug server and the GHS debug utilities.

Example:
None.

Command Line
None.

Workaround
A new tool update is available as CPDW9XNT-CDR-V85X-V516-PATCH03 containing all new features required for the extended trace tool (QB-V850E2-SP), suitable only for IECUBE2. The patch is available at

http://www.renesas.eu/updates?id=26

Please select the actual patch update:
CPDW9XNT-CDR-V85X-V516-PATCH04zip

No. a84	***gsrec is missing last segaddr record***

Version Information

V5.1.6C

Details:
GSREC missing the last segaddr record in Intel HEX format.

Example:
None.

Command Line
None.

Workaround
Please use new release V5.1.7D

No. a85	***Internal error of linker elxr***

Version Information

V5.1.6C

Details:
Under some circumstance the linker elxr produces an internal error.

Example:
Not available.

Command Line
None.

Workaround
Please use new release V5.1.7D

No. a86	***Gasmlist fails to display prolog and epilog***

Version Information

V5.1.6C

Details:
The utility “gasmlist” is failing to display prologue and epilogue for routines compiled with DWARF debug information.

Example:
None.

Command Line
None.

Workaround
None.
Please use new release V5.1.7D

No. a87	Compiler does not generate ".rozdata" section

Version Information

V5.1.6C

Details:
The compiler does not make use of ".rozdata" section, when it is used with the option "-no_rosda" or "-reserve_r5". The constant variable is located in '.rodata' section without further notice.

Example:
```
#pragma ghs startzda
const char id[]="My far Lady";
#pragma ghs endzda
```

Command Line
ccv850 -cpu=v850e -no_sda -reseve_r5

Workaround
None.
Please use new release V5.1.7D

No. a88	Assembler cannot jump to absolute address

Version Information

V5.1.6C

Details:
The assembler cannot jump to absolute address, if it is declared as an absolute address.

Example:
```
#define ADDRESS 0x8000
      jr    0x8000      -- fails
      jr    ADDRESS     -- fails
```

Command Line
ccv850 -cpu=v850e2v3

Workaround
Do not use absolute addresses.
Declare the absolute jump target using '.set' instruction, for example

```
.set ADDRESS 0x8000
      jr    ADDRESS           -- OK
      jr    ADDRESS + offset  -- OK
```

Please use new release V5.1.7D

No. a89	GHS target agent for external flash loading and memory test fails

Version Information

V5.1.6C, V5.1.7D

Details:

The utilities of the Multi debugger for external flash programming (gflash) and the memory test suite are not functional, if the target agent is used.

Example:

```
MULTI> memtest 0xfedc5000 0xfeddffff -size=4 -test=a0 -test=a1 -
tgtagent -tgtagentloc=0xfedc0000
```

Response

```
Loading executable: C:\Tools\GHS\v800517D\lib\v850\memtest...
Source Root:      C:\Tools\GHS\v800517D
Arguments: -test=3 0xfedc5000 0xfeddffff -pattern=1431655765 -size=4 -
minsize=1024 -maxerr=10 -repeat=1 -options=524288
Finished loading.
Downloading memory test target agent code to 0xfedc0000 and data to
0xfedc3650
multi: warning: Unable to find TCS_ENTITY for thread on target.
Memory test failed.
```

Workaround

Please use release V5.1.7D Patch01 or later from

http://www.renesas.eu/updates?id=26

No. a90	Internal CodeFactor Error

Version Information

V5.1.6C

Details:
The linker generates an error:
[elxr] (assertion) Internal CodeFactor Error

Example:
None.

Command Line
None.

Workaround
1. Use latest Patch Update for V5.1.6C, or
2. Please use release V5.1.7D Patch03
from

http://www.renesas.eu/updates?id=26

No. a91	Delayed/broken start of MULTI IDE

Version Information

V5.1.6C,V5.1.7D

Details:
Under Win7 (32bit/64bit) it was possible that MULTI did not always properly start up after invocation.

Example:
None.

Command Line
None.

Workaround
Please use release V5.1.7D Patch03 or later from

http://www.renesas.eu/updates?id=26

No. a92	Option '-list' influenced code optimizer

Version Information

V5.1.6C,V5.1.7D

Details:
When the compiler is used with option '-list', the code is differently optimized than without that option.

Example:
None.

Command Line
ccv850 –cpu= v850e2v3 –list –O test.c

Workaround
Please use release V5.1.7D Patch03 or later from

http://www.renesas.eu/updates?id=26

No. a93	Option '-misalign_pack' generates optimized target access

Version Information

V5.1.6C,V5.1.7D

Details:
When the option '-misalign_pack is used, the compiler generates target access modes, which are not allowed in volatile address spaces.

Example:

```
typedef union {
      struct {
            uint8 u81;
            uint8 u82;
            uint16 u161;
            } st;
      uint16 u162[2];
} T_s1, *T_ps1;
```

The widest access to an object of type T_s1 cannot be be wider than the biggest data type used within the object. In this example, it may be max. 16-bit access only. This rule applies only to objects without the attribute 'volatile'.

Command Line
None.

Workaround
Please use release V5.1.7D Patch03 or later from

http://www.renesas.eu/updates?id=26

No. a94	ELXR internal error

Version Information

V5.1.6C,V5.1.7D

Details:
Sometimes, the linker generates randomly an internal error blaming codefactor option, even if this option is not used at all.

Example:
None.

Command Line
None.

Workaround
Please use release V5.1.7D Patch03 or later from

http://www.renesas.eu/updates?id=26

No. a95	ELXR internal error with option '-v'

Version Information

V5.1.6C,V5.1.7D

Details:
Sometimes, the linker generates randomly an internal error blaming codefactor option, even if this option is not used at all.

Example:
None.

Command Line
None.

Workaround
Please use release V5.1.7D Patch03 or later from

http://www.renesas.eu/updates?id=26

No. a96	Simulator is changed for SCH1L instruction simulation

Version Information

V5.1.6C,V5.1.7D

Details:
The simulator did not implement correctly the instruction SCH1L.

Example:
None.

Command Line
None.

Workaround
Please use release V5.1.7D Patch03 or later from

http://www.renesas.eu/updates?id=26

No. a97	Missing Compiled Line

Version Information

V4.2.4

Details:
The code at line 139 of the sample module is not properly translated, when a combination of the options "-sda=all -Osize -shortenum" is used.

Example:
The register r12 is not initialized

```
              220 --138: {
              221 --139:   if (BINARY_ACTIVE == (TBinary)GetVSG())
              222 --line139
              223
              224 .LDWlin8:
--missing:    225       ld.b  sdaoff(_VSG)[gp],r12
00000034 8162 225       shr   1,r12
00000036 f905 226       bnl   .L132
              227 --140:                          {
              228 --141: VsgF_Temp = SWITCH_UP;
              229 --line141
```

Command Line
Any combination of "-sda=all -Osize -shortenum".

Workaround
Please use release V5.x.x or higher.
The workaround is to use the options '-Z2878' along with '-Z1710'. Together these should avoid the optimization causing this behavior.

No. a98	ELXR generates different output

Version Information

V5.1.6C,V5.1.7D, V5.3.0

Details:
The linker result is sometimes different on various host systems. The code is allocated in different order (different addresses), but not changed in the functionality.

Example:
A sample cannot be given, as the code size and structure of the application is very individual.

Command Line
ccv850 @ld_opt.xcl –Olink -o test.elf

Workaround
Do not use the linker option '-Olink", or
Turn off the icacheizer optimization manually by using the option '-no_icachizer'.
If '-Olink' was used before, it shall be replaced at least by the options

- '-delete' and
- -shorten_loads.

No. a99	Address Assignment set to 0 with pragma ghs io

Version Information

V5.1.7D

Details:
The assignment of an address of a variable which has been declared with "#pragma ghs io var" will be set to value 0. Prerequisite is that the assignment is placed in different C-Module than the reference to the variable (see example).

Example:

```
C-Module 1:

extern volatile unsigned short* port_register_0;
volatile unsigned short * address_2;

void main (void)
{
   address_2 = port_register_0;
}


C-Module 2:

#include "df3580_0.h"

volatile unsigned short* port_register_0 = &PMC0;
```

Command Line
ccv850 –cpu=v850e3v5 @df3358.ld –O test.c –o test

Workaround
1. Please use version V5.3.0, or
2. Update with latest patch for V5.1.7D, Patch04 or later from

http://www.renesas.eu/updates?id=26

No. a100	V61.4 release with incomplete E1 debugger support

Version Information

V6.1.4

Details:
The basic installation of the tools will NOT include all required support files for all E1 debugger operations.

Example:
None.

Command Line

Workaround
Please use latest V6.1.4 update CD Image with compiler 2012.5.5.

No. a101	Internal Compiler Error “delptr”

Version Information

V6.1.4/2012.5.5

Details:
The compiler drivers may show an internal compiler error, if linker is invoked. The message appears like this:

```
ccv850: Warning: Internal error on ..\src\builder\call.c, line
760: delptr=0xde3398 > 0xde2a30+0x259
```

Example:
None.

Command Line
Ccv850 –rh850 –O –o test.out *.o

Workaround
Please use latest patch Y-GHS-MULTI-V800-FULL-V614_2012.5.5-PATCH01

No. a102	Bad short load/store optimization

Version Information

V6.1.4/2012.5.5

Details:
There is a possibility for bad short load/store optimization when a “constant address + fixed offset” were dereferenced. It only affects addresses that fit within the offset of a short load/store instruction, and only when the additional fixed offset would place it outside that range, and even then only in limited circumstances.

Example:
(Over)simplified:

```
*((int*)(0x00000010 + 16)
```

Command Line
Ccv850 –rh850 –O –o test.out *.o

Workaround
Please use latest patch Y-GHS-MULTI-V800-FULL-V614_2012.5.5-PATCH01

No. a103	Size optimization of bitfields

Version Information

V6.1.4/2012.5.5

Details:
The size optimization is improved using a bit-field inside union

Example:

```
if () A[u.i].i = 1; else A[u.i].i = 0;
```

Command Line
Ccv850 –rh850 –O –o test.out *.o

Workaround
Please use latest patch Y-GHS-MULTI-V800-FULL-V614_2012.5.5-PATCH01

No. a104	Compiler may hang

Version Information

V6.1.4/2012.5.5

Details:
The compiler may hang when compiling a very particular kind of loop. In this case, an object module is not generated.

Example:
None.

Command Line
Ccv850 –rh850 –O –o test.out *.o

Workaround
Please use latest patch Y-GHS-MULTI-V800-FULL-V614_2012.5.5-PATCH01

No. a105	Error in gsrec

Version Information

V5.1.7D

Details:
Conversion with utility program gsrec might fail, when converting to intel hex format with padding option enabled.
Padding of holes is done incompletely and data is lost.

Example:
None.

Command Line
gsrec.exe test.elf -h -hex386 -bytes 32 -pad1 0x00000000 0x00FDFFF 0x00 -o test.hex

Workaround
1. Please use gsrec version V5.3.0 or later
2. Please use latest patch http://www.renesas.eu/updates?id=26

No. a106	Linker does not delete all unused functions

Version Information

V6.1.4 all versions

Details:
Along with the introduction of the assembler switch statement in rh850 cores, the linker was not able to recognize unused references.

Example:
None.

Command Line
ccv850 –cpu=rh850 –O –o test.out –delete *.o

Workaround
For V6.1.4/2013.5.5, please use latest patch
Y-GHS-MULTI-V800-FULL-V614_2013.5.5-PATCH02

No. a107	Problem with hex constants in ease850

Version Information

V6.1.4 all versions

Details:

Example:
None.

Command Line
ccv850 –cpu=rh850 –O –o test.out –delete *.o

Workaround
For V6.1.4/2013.5.5, please use latest patch
Y-GHS-MULTI-V800-FULL-V614_2013.5.5-PATCH02

No. a108	***Compiler interrupt routines do not save FPSR and FPEPC***

Version Information

V6.1.4 all versions

Details:
P10:
BTO-1906: Interrupt services implemented with the compiler do not save FPU related status register FPSR and FPEPC.

Example:
None.

Command Line
ccv850 –cpu=rh850 –O –o test.out

Workaround
For V6.1.4/2013.5.5, please use latest patch
Y-GHS-MULTI-V800-FULL-V614_2013.5.5-PATCH02

No. a109	Linker invokes unknown section SHN_ABS

Version Information

V6.1.4 all versions

Details:
When inline assembler of the compiler is using a jump relative instruction, such as

```
asm (" jr   _START_TIMERM") ;
```

The linker “elxr.exe” claims for unknown section SHN_ABS.

Example:
None.

Command Line
ccv850 –cpu=rh850 –O –o test.out

Workaround
For V6.1.4/2013.5.5, please use latest patch
Y-GHS-MULTI-V800-FULL-V614_2013.5.5-PATCH02

No. a110	***Multi IDE crash in New Project Wizard with MultiCoreArchive projects***
	Version Information *V6.1.4 all versions* *Details:* Sometimes, the NPW crashes, if a new object is inserted in the Multi project manager. *Example:* None. *Command Line* Multi.exe *Workaround* For V6.1.4/2013.5.5, please use latest patch Y-GHS-MULTI-V800-FULL-V614_2013.5.5-PATCH02

No. a111	Invalid bitfield handling in optimized If/else clause

Version Information

V5.1.7D, V2012.x, V2013.1.x

Details:
The broken optimization combines assignments to bitfields so that a single load-modify-store sequence can perform multiple assignments. The issue involves an incorrect interpretation of control flow.

Example:

```
extern int printf(const char *, ...);
struct {
   unsigned int b1:1;
   unsigned int b2:1;
   unsigned int b3:1;
   unsigned int b4:1;
   unsigned int b5:1;
} s = { 0 };

unsigned char flag=0, count=0 ;

void func(void) {
    if (flag == 3) {
      s.b4 = 0;
      if (count == 1)
         flag = 2;
   } else
      s.b1 = 0;
   s.b5 = 1;
}

int main( void) {
    func();
    if (s.b5 != 1)
        printf("%s: b5 is false\n", __FILE__);
    return 0;
}
```

Command Line
ccv850 -cpu=v850e2 -G –Ospace test.c

Workaround
For V6.1.4/2012.5.5, please use latest patch
Y-GHS-MULTI-V800-FULL-V614_2012.5.5-PATCH01

For V6.1.4/2013.1.5, please use latest patch
Y-GHS-MULTI-V800-FULL-V614_2013.1.5-PATCH02

In case of V5.x.x:
The optimization limitation can be workaround by using special compiler option -Z3245 or -Z990
In general, the limitation can be removed, if the variable is implemented/declared with ‘volatile’ attribute.

No. a112	P26: TOOLS-20948 write to global variable was wrongly moved before __DI

Version Information

V6.1.4/2013.5.5

Details:

TOOLS-20948 write to global variable was wrongly moved before __DI

The problem here is that the write to variable “Glob” between the DI/EI pair is moved outside the DI/EI by the codemotion optimization.

Example:

```
#include <stdio.h>
volatile int Flag=1;
volatile int Glob=0;
int a = 1;

int main() {
    if (Flag==1) {
        __DI();
        Glob=a;
        __EI();
    } else
        Glob=a;

    printf("Hello world. %d \n", Glob);
    return 0;
}
```

Command Line

ccrh850 –cpu=rh860 –c -O

Workaround

For V6.1.4/2013.5.5, please use latest patch
Y-GHS-MULTI-V800-FULL-V614_2013.5.5-PATCH02
plus v800-v2013.5.5-comp-P26 or later

No. a113	P23: BTO-2134 fix error in code motion optimization

Version Information

V6.1.4/2013.5.5

Details:

BTO-2134 fix error in code motion optimization

An optimization which moves code outside of a conditional, such as “if-then-else”, is performed in cases where it is not entirely safe, resulting in incorrect behaviour in very rare cases. Below is reduced from the example originally reported by the customer.

Example:

```
void test( int flag, unsigned char start) {
    unsigned char u;
    for ( u=start; u<15; u++ )
        if ( flag )
            Display(u,u+1);
        else
            Other(u,u+1);
}
```

Command Line

ccrh850 –cpu=rh860 –c -O

Workaround

For V6.1.4/2013.5.5, please use latest patch v800-v2013.5.5-comp-P26 or later

No. a114	P26: BTO-2144 update global loop variables in nested loops

Version Information
V6.1.4/2013.5.5

Details:
BTO-2144 update global loop variables in nested loops;
An optimization for simple, nested loops did not check for the case where one or both of the loop control variables are defined outside of the function and the inner loop contains a function call which might use the variables. The global loop control variables were not updated within the loop and therefore the values were wrong when the function was called.

Example:

```
unsigned char i, j;
unsigned char array[32][32];
void func(void) { array[origi][origj] ++; }

void test(void) {
    for (i = 0; i < 8; i ++)
        for (j = 0; j < 16; j ++)
            func();
}
```

With simple, nested loops something very close to the code above is meant.

Command Line
ccrh850 –cpu=rh860 –c -O

Workaround
For V6.1.4/2013.5.5, please use latest patch
v800-v2013.5.5-comp-P26 or later

No. a115	P25: BTO-2136/ .ghsexports with library; BTO-2139 exportall should not imply extractall

Version Information

V6.1.4 all versions

Details:
P25 gcores update:
BTO-2136: now works properly along with ".ghsexports"- file using a library;
BTO-2139: the exportall option does not imply extractall

Example:
None.

Command Line
gcores core1 core2 share –o allcores

Workaround
For V6.1.4/2013.5.5, please use latest patch
v800-v2013.5.5-comp-P25

No. a116	P22: BTO-2095 ease850 now recognizes bt and bf

Version Information

V6.1.4 all versions

Details:
P22 assembler update:
BTO-2095, the assembler ease850 now recognizes bt and bf instructions.

Example:
None.

Command Line
ease850 test.850

Workaround
For V6.1.4/2013.5.5, please use patch
v800-v2013.5.5-comp-P22

No. a117	P21: BTO-2089 accurate simulation of “ldsr FPST” and “ldsr FPSFG”

Version Information

V6.1.4 all versions

Details:
P21 simulator update simrh850 contains multiple issues:
BTO-1935 corrected simulation of expression “0x80000000 % -1“.
BTO-2005 simrh850 ignored software breakpoint when no .syscall section in program.
BTO-2089 accurate simulation of “ldsr FPST” and “ldsr FPSFG”.

Example:
None.

Command Line
MULTI> connect simrh860 –cpu=rh850g3m test.out

Workaround
For V6.1.4/2013.5.5, please use patch
v800-v2013.5.5-P21

(C) Valid Specification

Multi2000 V3.x:

Item	Date published	Document No.	Document Title
1	October 2001	10102	Getting Started with MULTI 2000, v3.5
2	October 2001	10097	Green Hills Installation and Licensing Guide, v3.5
3	October 2001	10075	MULTI 2000 User´s Guide, v3.5
4	January 2002	10118	Embedded V800 Development Guide, v3.5
5	September 2001	10072	Green Hills C/C++ User´s Guide, v3.5
6	March 2002	10138	Target Connection User´s Guide for V800, v3.5

Multi2000 V4.x.x:

Item	Date published	Document No.	Document Title
1	March 2004	10233	Getting Started V4.0
2	January 2004	10169	Building Applications for Embedded V800 V4.0
3	April 2004	10211	Editing Files and Configuring the IDE V4.0
4	March 2004	10249	Configuring Connections for V800 Targets V4.0
5	April 2004	10212	Debugging V4.0

Multi2000 V5.x.x:

Item	Date published	Document No.	Document Title
1	December 2009	10390	Getting Started
2	May 2010	10415	Building Applications for Embedded V800
3	December 2009	10406	Editing Files and Configuring the IDE
4	May 2010	10427	Configuring Connections for V800 Targets
5	May 2010	10630	Debugging
6	May 2010	10631	Debugging Command reference

Multi2000 V6.x.x:

Item	Date published	Document No.	Document Title
1	April 24, 2012	start-463286	Getting Started
2	July 7, 2012	build_v800-468493	Building Applications for Embedded V800
3	April 24, 2012	edit-463286	Editing Files and Configuring the IDE
4	July 7, 2012	connect_ v800-468493	Configuring Connections for V800 Targets
5	April 24, 2012	debug-463286	Debugging
6	April 24, 2012	debug_cmd-463286	Debugging Command reference

Multi2000 V6.x.x/2013.5.5:

Item	Date published	Document No.	Document Title
1	April 24, 2012	start-463286	Getting Started
2	October 4, 2013	build_v800-496213	Building Applications for Embedded V800
3	April 24, 2012	edit-463286	Editing Files and Configuring the IDE
4	October 4, 2013	connect_v800-496213	Configuring Connections for V800 Targets
5	August 29, 2012	debug-471430	Debugging
6	April 24, 2012	debug_cmd-463286	Debugging Command reference

(D) Revision History

Item	Date published	Document No.	Comment
1	04-01-2002	DTOP0010V10	First release
2	07-23-2002	DTOP0010V20	Second release
3	09-12-2002	DTOP0010V21	Items a11-a15 added
4	01-27-2003	DTOP0010V22	Items a16-a20 added
5	04-10-2003	DTOP0010V23	Items a21-a22 added
6	08-27-2003	DTOP0010V24	Items a23-a25 added
7	03-08-2004	DTOP0010V25	Items a26-a27 added
8	03-22-2004	DTOP0010V26	Items a28-a29 added
9	04-07-2004	DTOP0010V27	Item a30 added
10	05-06-2004	DTOP0010V28	Items a31-a32 added
11	24-08-2004	DTOP0010V29	Items a33-a37 added
12	08-10-2004	DTOP0010V30	Multi 4.0.5 added to version list
13	27-12-2004	DTOP0010V31	Items a38-a46 added
14	20-01-2005	DTOP0010V32	Multi 4.0.5C added to version list
15	21-04-2005	DTOP0010V33	Items a47-a50 added
16	22-08-2005	DTOP0010V34	Items a51-a52 added
17	23-08-2005	DTOP0010V35	Multi 4.07 added to version list
18	03-11-2005	DTOP0010V36	Item a52 corrected, Item a53- 54 added Modified version list: Multi 4.07A added, V3.3 removed
19	12-12-2005	DTOP0010V37	Items a53 to a55 corrected, Items a56, a57 added.
20	13-01-2006	DTOP0010V38	Items a58 to a62 added
21	20-04-2006	DTOP0010V39	Items a63 to a65 added
22	03-07-2006	DTOP0010V40	Item a66 added
23	03-08-2006	U18068EE4V0IF00	Change of document numbering system
24	11-08-2006	U18068EE5V0IF00	Item 67 added
25	21-09-2006	U18068EE6V0IF00	Item 68 added
26	22-12-2006	U18068EE7V0IF00	Item 69 added
27	16-03-2007	U18068EE8V0IF00	Items 70 to 72 added
28	05-09-2007	U18068EE9V0IF00	Item 73 and 74 added, changed Font types
29	21-09-2007	U18068EEAV0IF00	Item 75 added
30	11-07-2008	U18068EEBV0IF00	Item 56 modified
31	17-08-2008	U18068EECV0IF00	Compiler Update V4.2.4, Item 76
32	18-03-2009	U18068EEDV0IF00	Items a77 and a78 added
33	26-06-2009	U18068EEEV0IF00	Item a79 added
34	14-08-2009	U18068EEFV0IF00	Item a80 added, general rework of this document.
35	02-11-2009	U18068EEGV0IF00	Item a81 added.
36	29-06-2010	r20tu0003ed1800	Items a82 and a83 added, changed document number, new release from a84 to a87
37	29-07-2010	r20tu0003ed1801	Item a88 added
37	11-11-2010	r20tu0003ed1802	Item a89 added
38	14-12-2010	r20tu0003ed1803	Items a40 to a62 corrected in Chapter (A) Table Item a90 added.
39	12-01-2011	r20tu0003ed1804	Correction of broken Hyperlinks pointing to RENESAS Web space.
40	25-05-2011	r20tu0003ed1805	Items a91, a92, a93, a94, a95, a96

Item	Date published	Document No.	Comment
41	09-02-2012	r20tu0003ed1806	Items a97, a98 are added. Additional release status of V5.3.0 added to table.
42	29-08-2012	r20tu0003ed1807	Items a99 and a100 are added. Document changed to incorporate new order code and new GHS version.
43	14-12-2012	r20tu0003ed1808	Items a101, a102, a103 and a104 are added.
43	12-11-2013	r20tu0003ed1809	Items a105, a106 are added.
44	21-03-2014	r20tu0003ed1810	Items a107, a108, a109 and a110 are added.
45	29-04-2014	r20tu0003ed1811	Item a111 added, some general corrections in the document
46	16-09-2014	r20tu0003ed1812	Items a112, a113, a114, a115, a116 and a117 are added. Change in patch release procedure: Original GHS report numbers are now referenced.

Renesas Electronics Europe